Niveau(x) d'enseignement: 7 Numéro EDUQ : 8470830
Centre d'intérêt: 846

Lund, Margaret E.

Inquiry into Virtual Storage Access Method (VSAM) / by Margaret E. Lund. -- English version. -- [Westmount] : Collège Dawson, Data Processing Department, 1981.
v, 74 f. ; 28 cm.

"A report submitted to the Programme de subvention à l'innovation pédagogique, Service de l'enseignement collégial, Gouvernement du Québec". Bibliogr.: f. 74.

Descripteur(s): *Informatique *Ordinateur *Besoins de formation *Elaboration de programmes *Collégial (Niveau)

Identificateur(s) de sujet: *Programmation informatique *VSAM *Virtual storage access method *Méthode d'accès en mémoire virtuelle *Mémoire virtuelle

---

L'informatique de gestion utilise de plus en plus la méthode d'accès en mémoire virtuelle (Virtual storage access method, ou VSAM). Il s'agit d'une méthode de conservation et d'extraction de fichiers informatisés conçue pour les ordinateurs du système 370 d'IBM. L'étude vise à faire le point sur le VSAM et à déterminer si son enseignement doit être offert dans le cadre des cours d'informatique de niveau collégial.

Après avoir traité des avantages du VSAM, le rapport décrit la programmation avec cette méthode et l'application qu'on peut en faire sur l'ordinateur de SIME9 qui est à la disposition du réseau collégial. Il traite ensuite de l'utilisation du VSAM amorcée dans cinq grandes entreprises de la région de Montréal qui engagent des diplômés en informatique de niveau collégial. La dernière partie contient des recommandations quant à l'enseignement du VSAM au niveau collégial.

Etant donné que le VSAM est une méthode à laquelle IBM accorde une grande priorité et qui devrait donc se repandre dans l'industrie, il apparaît nécessaire d'en tenir compte dans la formation des futurs programmeurs. Par conséquent, le rapport recommande que partout où l'entraînement se fait sur de l'équipement IBM, on prenne les dispositions nécessaires en vue d'inclure la formation au VSAM et aux notions sur lesquelles il repose.

Type de document: r
Financement: Bénéficiaire - Collège Dawson
Organisme(s) source(s) - MEQ-DGEC
Programme(s) de subvention - PROSIP

EDUQ 84/08410

LUND,MARGARET E.

Inquiry into Virtual Storage
Access Method (VSAM)

EDUQ 8470830

Inquiry into Virtual Storage Access Method (VSAM)

by

Margaret E. Lund

A report submitted to the
Programme de Subvention à
l'innovation pédagogique, Service
de l'enseignement collégial,
Gouvernement du Québec.

English Version

Data Processing Department
Dawson College
Montreal, Quebec

June, 1981

TABLE OF CONTENTS

	Page
Table of Contents	ii
Preface	iv
CHAPTER I - Advantages of using VSAM and rationale for change	1
CHAPTER II - Programming using VSAM and adaptations to SIMEQ	4
CHAPTER III - Report on discussions with industry on the use of VSAM	9
CHAPTER IV - Recommendations on the teaching of VSAM	13
Appendix A - Sample VSAM File Processing Program COBOL ESDS Create and Extend COBOL ESDS Update and Extend	17
Appendix B - Sample VSAM File Processing Program COBOL KSDS Create COBOL KSDS Update IDCAMS Creation of Alternate Index over KSDS COBOL Alternate Index Reference on KSDS	23
Appendix C - Sample VSAM File Processing Program COBOL RRDS Create COBOL RRDS Sequential Update COBOL RRDS Random Retrieval	31
Appendix D - Sample VSAM File Processing Program PL/I ESDS Create and Extend IDCAMS Creation of Alternate Index over ESDS PL/I Alternate Index reference of ESDS	44
Appendix E - Sample VSAM File Processing Program PL/I KSDS Create PL/I KSDS Update	52

	Page
Appendix F - Sample VSAM File Processing Program PL/I RRDS Create PL/I RRDS Update	60
Appendix G - Sample VSAM File Processing Program Assembler ESDS Create and Extend	68
Appendix H - Sample VSAM File Processing Program Assembler KSDS Create (Variable-length records)	71
Bibliography	74

PREFACE

Discussions with data processing professionals in the companies that employ graduates from the three-year career program in Data Processing in the CEGEPs (College de l'enseignment général et professionnale), the system of post-secondary colleges in the province of Quebec, indicate that among users of IBM computing systems, Virtual Storage Access Method (VSAM) is being introduced into business data processing operations.

The purpose of this study is to consider the need for a change to VSAM, to demonstrate its use on the IBM computer which supplies computing services to the CEGEPs, and to determine to what extent the operations in some Montreal computing installations have been converted to VSAM processing, in order to make recommendations regarding changes that could be made to the Data Processing career program.

This report, which is available in both English and French versions, was first written in English and then the text was translated into French.

The sample programs which accompany this report will be a useful reference for anyone who is learning to program using VSAM. At the present time there are few programming language texts available which discuss VSAM adequately. The information which is needed to enable one to use VSAM data sets is distributed over a number of references, and

if examples are given, they are often incomplete, or else the various components of a job (JCL, IDCAMS and applications programs) are not brought together in the sequence in which they need to be executed.

The major funding for this project was provided by the Government of Quebec, Programme de subvention à l'innovation pédagogique, Service des programmes, Direction général de l'enseignment collégial. Dawson College, Montreal, provided additional funding through its Institutional Research Committee. Access to computer facilities at SIMEQ for testing the sample programs was provided by the Data Processing Department at Dawson College. Translation of the text from English into French was done by Suzanne Godin-Nash at Dawson College. The author is a teacher in the Data Processing Department at Dawson College.

CHAPTER I
Advantages of using VSAM and Rationale for Change

VSAM (Virtual Storage Access Method) is a method for storing and retrieving files on IBM System/370 computers running OS/VS1 or OS/VS2 (operating system/virtual storage, option 1 or 2).

The purpose of this study is to investigate VSAM and to make recommendations as to whether applications programmers should be trained to use it. It must be emphasized that VSAM is an IBM product for use on large IBM computers. It is also used on Amdahl 470 computers which operate completely under IBM control programs and simulate IBM System/370 computers. Any consideration of VSAM is only meaningful if programmers are to be trained on IBM equipment or its equivalent to use the methods for storing, retrieving and updating files on direct access storage devices. Other computer manufacturers will use their own methods for storing records in files, although the basic principles involved in the various file organizations will be similar from one computer system to another.

The non-VSAM IBM Access Methods which are in popular use today were first used on the IBM System/370 computers which became available in 1964 to handle the storing and retrieving of data in files on sequential access devices (e.g. magnetic tape units) and on direct access devices (e.g. magnetic disk units). Depending upon the storage

device used, and the nature and organization of the records to be stored (sequential, by index, or random), a choice is made between the five basic methods of storing and retrieving records, namely,

BSAM - Basic Sequential Access Method, which includes QSAM (i.e. Queued Sequential Access Method)

BDAM - Basic Direct Access Method

BISAM - Basic Indexed Sequential Access Method, which includes QISAM (Queued Indexed Sequential Access Method)

BPAM - Basic Partitioned Access Method

BTAM - Basic Telecommunications Access Method

All of the above Access Methods, with the exception of BTAM, are included within the courses taught in the Data Processing Program at Dawson College, Montreal.

However, with the introduction of the IBM System/370 series of computers in 1970, a greater variety of types of processing is possible, and new file storage devices can be used. In order to take advantage of the increased capabilities in a System/370 computer, a new Access Method has been introduced by IBM to handle more efficiently the varied demands made upon a computer installation today. VSAM is a single, versatile, high-performance Access Method for use with direct access storage devices which can replace a large proportion of the work processed under the earlier Basic and Queued Access Methods. There are similarities and functional equivalents between the two methods, e.g. an Indexed Sequential file in ISAM is similar to a key-Sequenced Data Set in VSAM. VSAM handles indexes, records and files more efficiently and makes

better use of storage (e.g. key compression).

The transition by industry to the new method can be gradual. The programs written using IBM System/360 methods continue to function on a System/370. However, the advantages (ease of use, a single common set of commands, more efficient storage methods and faster processing of records) make the change to VSAM worth serious consideration.

The name VSAM is used to describe both the Access Method and the types of data sets (or files) which it uses. VSAM uses three types of data sets.

ESDS - Entry-Sequenced Data Set
which can be used in place of SAM (Sequential) data sets.

KSDS - Key-Sequenced Data Set
which uses indexes to locate records and which replaces ISAM (Indexed) data sets.

RRDS - Relative-Record Data Set
which retrieves records by relative record number. It is similar to the earlier Relative Record organization.

There is no VSAM equivalent for the Relative Track organization which is available from COBOL programs.

CHAPTER II
Programming using VSAM and adaptations to SIMEQ

The use of VSAM access methods and data sets is demonstrated in the sample programs which appear in the Appendices. These programs have been tested at SIMEQ on an IBM System/370 Model 3033 running OS/VS2 MVS Release 3.7 and JES2 NJE Release 3.0. SIMEQ (Sérvice de l'information du Ministère de l'education du Québec) is the computing centre administered by the Education Department of the Province of Quebec. It provides computing services to the school boards and colleges in the province for administrative purposes, and to the CEGEPs (post-secondary colleges) for pedagogical purposes.

It is not the aim of this Chapter to produce a comprehensive guide to using VSAM. This knowledge can be acquired by reading the appropriate manuals. A list of the IBM manuals which were used in the preparation of the sample programs appears in the Bibliography. However, a brief summary of the steps involved is given here, with notes on how to adapt these to jobs which are executed at SIMEQ. It is assumed in this discussion that the reader is already familiar with IBM System/360 or 370 Operating Systems.

In order to process VSAM data sets they must first be defined using the IBM Utility Program IDCAMS. This Utility Program is used to handle all the system functions for VSAM files e.g. to define files, to load, copy, delete or print them, to define and build alternate indexes, and to list the catalog entry for files. These functions are described

in the IBM manual "OS/VS2 Access Method Services", GC26-3841.

The commands to be executed by Access Method Services to enable VSAM files to be processed, in the order in which they must be executed, are as follows:

1. DEFINE MASTERCATALOG
2. DEFINE USERCATALOG
3. DEFINE SPACE
4. DEFINE CLUSTER
5. DEFINE ALTERNATE INDEX and DEFINE PATH

1. DEFINE MASTERCATALOG

is a privileged command. The master catalog at SIMEQ is already defined. It is allocated when the system is initialized.

2. DEFINE USERCATALOG

is normally optional. VSAM data sets must be catalogued in either the Master Catalog or a User Catalog. However VSAM files for users at SIMEQ must be catalogued in a User Catalog. SIMEQ has already defined User Catalogs and the DEFINE USERCATALOG command is invalid for users. The sample programs included in this study have been placed in User Catalog EDUC05.MASTRGUY. The catalog name was obtained by an individual request to SIMEQ. At SIMEQ the User Catalog must be named in creating the files in order that they may be subsequently retrieved. The name must also be supplied whenever the files are to be retrieved or updated.

3. DEFINE SPACE

allocates VSAM data space. To execute the sample programs a request was submitted to SIMEQ for space to be made

available to store the VSAM files. SIMEQ asked that these files have a 'V' in Position 1 of the Data Set Name and the user number in Positions 2, 3, and 4. This appears as 'V818' in all the DSnames in the sample programs. Space was assigned on the Volume EDUC05 by SIMEQ. Users themselves may not use the DEFINE SPACE command at SIMEQ. It is an invalid operation.

4. DEFINE CLUSTER

defines a VSAM data set. This command must be executed before the data can be placed on the file. SIMEQ requires that the attribute UNIQUE always be included in the DEFINE CLUSTER command. This means that the VSAM file is the only component in that data space. The space used for that data set may not be shared with other VSAM components (clusters, alternate indexes, etc.).

5. DEFINE ALTERNATEINDEX and DEFINE PATH

commands are needed if alternate indexes are used to reference the VSAM files.
An additional command, BLDINDEX, is required once the file has been created, before the file can be referenced using a secondary key.

In summary then, the sample programs in this study use the last two commands only, the first three being privileged commands. In addition, in order to perform routine operations on VSAM data sets, other Access Methods Services commands have been demonstrated, e.g. PRINT, LISTCAT, DELETE.

Following the definition of the file and its space

allocation, applications programs may then be written in COBOL, PL/I or OS/VS Assembler to create or update the files. In jobs involving a number of job steps the User Catalog name may be supplied for all the steps in a job by the use of a JOBCAT JCL statement placed before the first job step.

The programs in the Appendices illustrate the basic programming techniques for each type of file organization. Examples are presented in COBOL, PL/I and Assembler language covering the creation, access and update of VSAM files. Their purpose is to provide complete and working examples which illustrate the connections between JCL, IDCAMS commands and the applications programs, and to show the input/output statements associated with each file type. The actual manipulation of the data within the programs is intentionally limited and there has been little effort made to optimize the use of storage. The aspects of VSAM which have been explored concentrate on those tasks that graduates of a data processing program taking up positions as applications programmers in industry need to be familiar with, in order to work with VSAM.

Several of the programs have been adapted from sample programs in the appropriate IBM manuals, the COBOL KSDS program was modified from an example in the reference text "The COBOL Environment" by R.T. Grauer and M.A. Crawford, (Prentice-Hall, 1979), and the remainder were written by the author. In most cases structured programming techniques have been employed, and many of the programs were completely rewritten from the suggested examples, in order to conform to structured programming standards.

All of the programs have been computer tested at SIMEQ. The complete source deck, exactly as submitted for execution, is shown for each group of programs, followed by the results obtained from execution. These results show the reports generated by the applications programs, and the listings of the VSAM files output from the IDCAMS 'PRINT' command.

CHAPTER III
Report on Discussions with Industry on the use of VSAM

Five companies which employ graduates from the Data Processing programs in the Montreal area were approached in order to determine to what degree their operations had been converted to VSAM.

The representatives of the companies contacted could not always respond with exact answers to all the questions posed them, due to the fact that large companies have many different computer applications and it is not always possible for one individual to be familiar with all aspects of their company's data processing operations. However, even from this small sample the trend to VSAM can be seen.

The five companies approached were:

Alcan
Canadian National
Bank of Montreal
Royal Bank, and
Standard Life Insurance

At Alcan Dianne McDonald, Data Processing Supervisor, indicated that half of their files used Indexed file organization and the other half, Sequential. Of the existing programs about 20% used VSAM files and 80% non-VSAM. They use an Amdahl 470 V6 running under MVS. Current policy is to use only VSAM files (with the possible exception of Data Base). About 75% are KSDS and 25% ESDS. COBOL, Assembler, PL/I and alternate indexes are all used with VSAM. IDCAMS

commands are also used to handle non-VSAM files. The company began using VSAM a year ago and Dianne recommended that trainee programmers learn to use it.

At the Bank of Montreal Wayne Fowler in charge of User Technical Support and Data Management functions in Toronto offerred valuable insights into VSAM. The Bank is using VSAM in only one system out of about 60 or 70 systems. They are not satisfied with VSAM at the present time. The way it handles storage space and the catalog structure need improvement. The system allows programmers too much freedom and from a data management point of view this is not good, since the bank prefers to centralize control of data management. There are no plans to convert existing systems to VSAM as there is too much invested in existing systems to want to change. IBM is presently working on a rewrite of VSAM and two new Data Facilities packages should solve many of the problems encountered. The Bank uses four Amdahl 470 V8 computers and an IBM 3033. The emphasis in programming is upon PL/I. The Bank uses its own direct access method and in place of ISAM it uses IAM (Innovation Access Method). Plans call for a slight increase in VSAM in the next year rising to about 20% of all files in about three years time. It is recommended that programmers learn to use VSAM.

At Canadian National they have an IBM 3032, 3033 and an Amdahl 470 V5 running MVS. About 15 - 20% of all new programs use VSAM files, namely ESDS and KSDS. Programming is in COBOL and Assembler. Some of the files are being converted for use with IDMS. They have encountered problems with VSAM.

especially in handling catalogs with on-line users. There are performance problems in using alternate indexes. IDCAMS is not used to handle non-VSAM files. It is recommended that programmers be able to use both VSAM and the earlier Access Methods.

At the Royal Bank Greg Pruden, Project Manager, and Michael Gaudreau, DB/DC Support Specialist, offered the following comments on VSAM. VSAM is used directly for handling the VISA charge-card system in on-line and batch processing modes in a "crude Data Base fashion". An interface module for this system takes care of the Input/Output functions and relieves programmers of the need to define files. The bank is a "COBOL shop". KSDS and ESDS are used. VSAM is also used in conjunction with IMS, although FOCUS DBMS is also used. Data Base Management Systems do much of the processing so that the use of VSAM is transparent to many users. Many systems are being converted to VSAM. The interface to convert from ISAM to VSAM is used. MVS/SP operating system is used and problems have been encountered with the catalog under VSAM regarding the single ownership of a volume. IBM announced last year a new catalog structure under VSAM and it is expected that this will solve many problems. Since IBM's general direction is towards VSAM it is recommended that programmers have a background in VSAM.

Peter Simpson, Systems Analyst at Standard Life Insurance indicated that 95% of all files used indexed organization and 5% sequential. Of the existing programs, 70% use VSAM files and 30% non-VSAM. All new systems use only VSAM files, namely

ESDS and KSDS. The applications programs are written in COBOL and they do not use a Data Base Management System. He requested that the training programs "stress VSAM". Within the company the programmer skills with VSAM are increasing. A year ago alternate indexes were not used, but today they are.

The experience of the Montreal companies appears to be in line with a survey based upon the software used on IBM Series 360, 370, 30 and 4300 CPUs in 751 computer sites in the United States, conducted by International Data Corporation, which has been reported on in the journal "Computerworld" as follows:

> "VSAM is the preferred access method on IBM's general-purpose mainframes, the report continued. Over 53% of the respondents chose this method, with ISAM managing to retain a share of just under 37%.
>
> However, when looking at specific CPU families, ISAM remained the choice of 360 users with 73.9% of this group running this software. In contrast, well over half of the 370 sites opted for VSAM, and some 65% of the 4331 users made the same choice, the report continued."

CHAPTER IV
Recommendations on the teaching of VSAM

Apart from this study, none of the Faculty at Dawson College has used VSAM data sets. The Access Methods Services Utility Program (IDCAMS) has only been used for handling non-VSAM files on a limited basis. Discussion with teachers in other CEGEPs reveals a similar situation. Only the earlier Access Methods are being taught.

The orientation of the training of programmers will vary from one college to another. The local needs of the communities which the colleges serve will have an influence upon the type of training which is given to programmers to prepare them for the local job market. Those CEGEPs in the larger communities will be more likely to be supplying programmers to the large corporations where file storage methods will be an important part of the work which they will be doing.

One point which emerged from the investigation into company activities was that the case for VSAM is not quite as straightforward as the sample batch programs might indicate. VSAM data sets are often used inconjunction with other systems such as data base management systems, teleprocessing, and query and on-line systems. In addition, some of these other systems have their own access methods and file organizations or else a separate interface program handles the VSAM input and output statements. On the other hand, the environment in which students are trained to program at Dawsom College is a batch one. It is recommended that students be given experience in

working with on-line systems.

The most significant point which emerged from the discussions with industry representatives was that, despite the fact that some companies were not altogether satisfied with VSAM performance, they all strongly recommended that programmers should be trained to use VSAM. What is not so clear, however, is the way in which this should be accomplished. The many new and valuable areas for study in the data processing career program, when added to the curriculum, place an increasing burden upon teachers and students. In order to add VSAM to the curriculum less emphasis must be placed upon other aspects of programmer training.

At the present time we are in a period of transition. It is not possible to replace teaching of the earlier access methods with the teaching of VSAM. There are still many systems being written which use the earlier methods, in addition to those which are already in use. To convert the existing non-VSAM systems to VSAM is an expensive job and none of the companies surveyed had plans to convert all their systems to VSAM at this time.

In its favour, VSAM is a very simple system to use. The JCL needed and the input/output commands are less clumsy than for non-VSAM files and non-IDCAMS Utilities. It is only necessary to learn one set of commands to handle all three types of data sets in VSAM. It is not difficult to train programmers who have already mastered non-VSAM programming to adapt the basic concepts to VSAM.

IDCAMS is a valuable Utility Program not just for VSAM functions. Non-VSAM files previously needed many different Utility programs, each with its own set of JCL, to handle routine operations, e.g.

```
IEBPTPCH or IEBISAM to list files
IEBGENER            to copy them
IEHPROGM            to rename, catalog or scratch them
```

whereas many of these same commands can be executed using IDCAMS. With one set of JCL a sequence of commands can be executed in one job step. Such commands as PRINT, REPRO, DELETE, ALTER, DEFINE NONVSAM, DEFINE GENERATIONDATAGROUP, DEFINE ALIAS and LISTCAT can be used for non-VSAM files. The syntax of the commands in IDCAMS is easier to code. The time spent learning to use IDCAMS can replace the time previously spent on the various Utilities. Once programmers can use IDCAMS, then the task of mastering VSAM will be made that much easier.

In considering the need for VSAM it is being marketed by IBM a company which holds a dominent position in computers today, and so any major change adopted by IBM deserves serious consideration to be included in the training of programmers.

VSAM was first used by IBM ten years ago, and today it has expanded capabilities and a performance record. IBM is placing its major emphasis on VSAM. The latest VS COBOL compiler from IBM is designed to handle the three types of VSAM data sets as the standard method. The non-VSAM files are only mentioned in an Appendix. New systems and software which IBM markets use VSAM rather than non-VSAM files. IMS, the IBM Data Base Management System makes use of the alternate

indexing feature of VSAM. Thus, the full capabilities of IMS can be used with VSAM files. All of this indicates a major commitment to VSAM by IBM, and users can be assured that if there are at present some problems in using VSAM, that IBM will be working with industry to resolve them.

The above factors mean that a change to VSAM, and thus a change in the skills needed by programmers, is inevitable for those using IBM Systems. It is mostly just a question of time for industry to adapt and for IBM to produce better data management and catalog handling.

It can be seen from the discussions with industry that the conversion to VSAM is well underway and that the need for VSAM programmers exists today. By the time the programmer training programs respond to the new demand, this need will have increased and graduates with a knowledge of VSAM will have an employment advantage.

In conclusion it is recommended that faculty in the data processing career programs begin immediately to acquire the necessary skills to enable VSAM to be included in the curriculum wherever the training includes IBM equipment and the earlier Access Methods. The major emphasis should be upon using Key-Sequenced Data Sets, alternate indexes and the COBOL statements needed to handle these, but as can be seen from the survey, a broader knowledge, as indicated in the sample programs, is suggested.

Appendix A - Sample VSAM File Processing Program
COBOL ESDS Create and Extend
COBOL ESDS Update and Extend

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ @QUE
//JOBCAT DD DSN=EDUC05.MASTRGUY,DISP=SHR
//* TO DEFINE AN ESDS FOR COBOL VSAM CREATE
//CRCOB  EXEC  PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 DEL V818LUND.COBESDS.CLUSTER PURGE
 DEFINE CLUSTER( -
   NAME(V818LUND.COBESDS.CLUSTER) -
   VOLUMES(EDUC05) -
   NONINDEXED -
   RECORDSIZE(37 37) -
   TRACKS(1 1)-
   FOR(2) -
   UNIQUE) -
  CATALOG(EDUC05.MASTRGUY)
//* COBOL CREATE AND ADD PROGRAM FOR ESDS
//COBPGM EXEC COBVSCL
//COB.SYSIN  DD  *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CRESDS.
       REMARKS.  SAMPLE PROGRAM USED FOR CREATE AND UPDATE
                 (WITH ADDITIONS ONLY) A VSAM ESDS.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
        SOURCE-COMPUTER. IBM-370.
       OBJECT-COMPUTER. IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT ES-FILE ASSIGN TO AS-ESMAST
           ORGANIZATION IS SEQUENTIAL
           ACCESS MODE IS SEQUENTIAL.
           SELECT CARD-FILE ASSIGN TO UR-S-SYSIN.
       DATA DIVISION.
       FILE SECTION.
       FD  CARD-FILE
           RECORD CONTAINS 80 CHARACTERS
           LABEL RECORDS ARE OMITTED
           DATA RECORD IS CARD.
       01  CARD.
           05  CD-ID                    PIC 9(10).
           05  CD-NAME                  PIC X(20).
           05  CD-BAL                   PIC 9(5)V99.
           05  FILLER                   PIC X(43).
```

```
       FD  ES-FILE
           RECORD CONTAINS 37 CHARACTERS
           LABEL RECORDS ARE STANDARD.
       01  ES-DATA-RECORD.
           05  ES-ID                       PIC 9(10).
           05  ES-NAME                     PIC X(20).
           05  ES-BAL                      PIC 9(5)V99.
       WORKING-STORAGE SECTION.
       01  READER-STATUS                   PIC X(3)  VALUE 'NO'.
           88  OUT-OF-CARDS                VALUE 'YES'.
       PROCEDURE DIVISION.
       MAIN-ROUTINE.
           OPEN INPUT CARD-FILE.
                EXTEND ES-FILE.
           PERFORM READ-AND-WRITE-ES UNTIL OUT-OF-CARDS.
           DISPLAY 'ES-PROCESSING-COMPLETE'.
           CLOSE CARD-FILE, ES-FILE.
           STOP RUN.
       READ-AND-WRITE-ES.
           READ CARD-FILE AT END MOVE 'YES' TO READER-STATUS.
           IF NOT OUT-OF-CARDS
              WRITE ES-DATA-RECORD FROM CARD.
//GO EXEC PGM=*.COBPGM.LKED.SYSLMOD
//SYSOUT DD SYSOUT=*
//SYSPRINT  DD  SYSOUT=*
//ESMAST DD DSN=VA18LUND.COBESDS.CLUSTER,DISP=SHR
//SYSIN  DD  *
1135768451FRIBOURG NICK D.     0001753
1196627496JURINVILLE ROLLAND   0016294
5564311062CHIAZZESE LUCIANO    1834615
//PRT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT INDATASET(VA18LUND.COBESDS.CLUSTER) CHAR
//UP EXEC PGM=*.COBPGM.LKED.SYSLMOD
//SYSPRINT  DD  SYSOUT=*
//SYSOUT DD SYSOUT=*
//ESMAST DD DSN=VA18LUND.COBESDS.CLUSTER,DISP=SHR
//SYSIN  DD  *
6611630261MALETTE ALBERT       7346183
8843961417LESSARD CLAUDE       0010050
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT INDATASET(VA18LUND.COBESDS.CLUSTER) CHAR
//UPDATE EXEC COBVSCG,PARM.COB='STATE,FLOW'
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. UPDATE-SEQUENTIAL.
       REMARKS.  THIS PROGRAM DOES A SEQUENTIAL UPDATE ON A SEQUENTIAL
           FILE.  WHEN THE UPDATE CARD-ID IS MATCHED WITH A MASTER-ID
           THE UPDATE CARD REPLACES THE MASTER RECORD IN THE MASTER
           FILE.  BOTH FILES ARE IN ASCENDING SEQUENCE BY ID NUMBER.
           IF THE CARD-ID IS GREATER THAN ANY MASTER-ID THEN THE CARD
           RECORD IS ADDED ON TO THE END OF THE MASTER FILE.
       ENVIRONMENT DIVISION.
```

```
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-370.
OBJECT-COMPUTER.  IBM-370.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN TO UR-S-SYSIN.
    SELECT MASTER-FILE ASSIGN TO AS-ESMAST.
DATA DIVISION.
FILE SECTION.
FD  CARD-FILE
    LABEL RECORDS OMITTED.
01  CARD-RECORD.
    05 CARD-ID                      PIC 9(10).
    05 CARD-NAME                    PIC X(20).
    05 CARD-BALANCE                 PIC 9(5)V99.
    05  FILLER                      PIC X(43).
FD  MASTER-FILE
    LABEL RECORDS ARE STANDARD.
01  MASTER                          PIC X(37).
WORKING-STORAGE SECTION.
77  PREVIOUS-MASTER-ID              PIC 9(10) VALUE ZERO.
01  MASTER-RECORD.
    05 MASTER-ID                    PIC S9(10) VALUE -1.
    05 MASTER-NAME                  PIC X(20).
    05 MASTER-BALANCE               PIC 9(5)V99.
01  OUT-OF-CARDS-FLAG               PIC XXX VALUE 'NO'.
    88 OUT-OF-CARDS                 VALUE 'YES'.
01  ID-ERROR-FLAG                   PIC XXX VALUE 'NO'.
    88 ID-ERROR                     VALUE 'YES'.
01  END-OF-MASTER-FILE-FLAG         PIC XXX VALUE 'NO'.
    88 END-OF-MASTER-FILE           VALUE 'YES'.
PROCEDURE DIVISION.
MAIN-ROUTINE.
    OPEN INPUT CARD-FILE,
         I-O MASTER-FILE.
    PERFORM READ-AND-UPDATE UNTIL OUT-OF-CARDS.
    CLOSE CARD-FILE, MASTER-FILE.
    STOP RUN.
READ-AND-UPDATE.
    READ CARD-FILE AT END MOVE 'YES' TO OUT-OF-CARDS-FLAG.
    IF NOT OUT-OF-CARDS PERFORM UPDATE-FILE.
UPDATE-FILE.
    IF NOT END-OF-MASTER-FILE
      MOVE 'NO' TO ID-ERROR-FLAG
      PERFORM READ-MASTER UNTIL MASTER-ID = CARD-ID
        OR ID-ERROR-FLAG = 'YES'
        OR END-OF-MASTER-FILE-FLAG = 'YES'
      IF NOT END-OF-MASTER-FILE AND NOT ID-ERROR
        REWRITE MASTER FROM CARD-RECORD.
    IF END-OF-MASTER-FILE
      WRITE MASTER FROM CARD-RECORD.
READ-MASTER.
    READ MASTER-FILE INTO MASTER-RECORD
      AT END MOVE 'YES' TO END-OF-MASTER-FILE-FLAG
             CLOSE MASTER-FILE
             OPEN EXTEND MASTER-FILE.
```

```
           IF NOT END-OF-MASTER-FILE
             IF CARD-ID IS > PREVIOUS-MASTER-ID AND < MASTER-ID
               MOVE 'YES' TO ID-ERROR-FLAG
               DISPLAY 'NON-EXISTING RECORD TO UPDATE ', CARD-ID.
           MOVE MASTER-ID TO PREVIOUS-MASTER-ID.
//GO.SYSDBOUT DD SYSOUT=*
//GO.SYSOUT DD SYSOUT=*
//GO.ESMAST DD DSN=V818LUND.COBESDS.CLUSTER,DISP=SHR
//GO.SYSIN DD *
1135768451FRIBOURG NICK D.     0001945
1459783562SMITH THOMAS         0006555
6611630261MALETTE ALBERTO K.   7346183
8891457632BONOMO FRANCESCA     7346183
9994641769DUFRESNE MICHELLE    0048229
//LIST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 LISTCAT -
   CAT(EDUCOS.MASTRGUY) -
   ENT(V818LUND.COBESDS.CLUSTER) -
   ALL
 PRINT INDATASET(V818LUND.COBESDS.CLUSTER) CHAR
 DEL V818LUND.COBESDS.CLUSTER PURGE
```

Contents of ESDS after Execution of
COBOL Create Program

LISTING OF DATA SET -[illegible]

RBA OF RECORD = 0
[illegible]

RBA OF RECORD = 37
[illegible]

RBA OF RECORD = 74
[illegible]

Contents of ESDS after Execution of
COBOL Extend Program

LISTING OF DATA SET -VS1.LONG.COBESDS.CLUSTER

RBA OF RECORD = 0
1132760451FR1BOURG WICK D. 0001735

RBA OF RECORD = 37
[illegible]

RBA OF RECORD = 74
[illegible]

RBA OF RECORD = 111
6O1.C3U261MALETTE ALBERT 7348163

RBA OF RECORD = 148
[illegible]

Report Produced by COBOL ESDS
Update and Extend Program

NON-EXISTING RECORD TO UPDATE 14597835e2

Contents of ESDS after Execution of
COBOL Update and Extend Program

LISTING OF DATA SET -V818LUNC.CORESDS.CLUSTER

RBA OF RECORD - 0
1135768451FRIBOURG NICK D. 0001945

RBA OF RECORD - 37
1196627496SINVILLE ROLLAND 0016294

RBA OF RECORD - 74
5564311062CHIAZZESE LUCIANO 1834615

RBA OF RECORD - 111

6611630261MALETTE ALBERTO K. 7346183

RBA OF RECORD - 148
8843961417LESSARD CLAUDE 0010050

RBA OF RECORD - 185

8891457632BONOMO FRANCESCA 7346183

RBA OF RECORD - 222
9994641769DUFRESNE MICHELLE 0048229

Appendix B - Sample VSAM File Processing Program
COBOL KSDS Create
COBOL KSDS Update
IDCAMS Creation of Alternate Index over KSDS
COBOL Alternate Index Reference on KSDS

SOURCE LISTING

```
*JOB CLASS=T
/*XEQ     eWOE
//JOBLAI DD DSN=EDUCUS.MASTRGUY,DISP=SHR
//* TO DEFINE A KSDS FOR COBOL VSAM CREATE
//DEF EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE VBIBLUND.COBKS.CLUSTER PURGE
 DEFINE CLUSTER -
    (NAME(VBIBLUND.COBKS.CLUSTER) -
    VOLUMES(EDUCUS) -
    INDEXED -
    RECORDSIZE(40 40)-
    REPLICATE -
    IMBED -
    CYL(1 1)-
    UNIQUE -
    KEYS(9  0) -
    FOR(3) -
    FSPC(15 10)) -
    DAT(EDUCUS.MASTRGUY)
//* COBOL PROGRAM TO LOAD THE DATA SET
//COBPGM EXEC COBVSCG,PARM='STATE,FLOW'
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID.   CREATE.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER.   IBM-370.
       OBJECT-COMPUTER.   IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT TRANSACTION-FILE ASSIGN TO UT-S-SYSIN.
           SELECT VSAM-FILE ASSIGN TO VSAMMAST
             ACCESS IS SEQUENTIAL
             ORGANIZATION IS INDEXED
             RECORD KEY IS VSAM-ID-NUMBER.
       DATA DIVISION.
       FILE SECTION.
       FD  VSAM-FILE
           LABEL RECORDS ARE STANDARD
           RECORD CONTAINS 40 CHARACTERS
           DATA RECORD IS VSAM-MASTER-RECORD.
       01  VSAM-MASTER-RECORD.
           05  VSAM-ID-NUMBER              PIC X(9).
           05  VSAM-NAME                   PIC X(25).
           05  VSAM-SALARY                 PIC X(6).
```

```
       FD  TRANSACTION-FILE
           LABEL RECORDS ARE OMITTED
           RECORD CONTAINS 80 CHARACTERS
           DATA RECORD IS TRANSACTION-RECORD.
       01  TRANSACTION-RECORD.
           05  TR-ID-NUMBER                PIC X(9).
           05  TR-NAME                     PIC X(25).
           05  TR-SALARY                   PIC X(6).
           05  FILLER                      PIC X(40).
       WORKING-STORAGE SECTION.
       77  WS-TRANSACTION-SWITCH           PIC X(3) VALUE 'NO'.
       PROCEDURE DIVISION.
           OPEN INPUT TRANSACTION-FILE
             OUTPUT VSAM-FILE.
           READ TRANSACTION-FILE
             AT END MOVE 'YES' TO WS-TRANSACTION-SWITCH.
           PERFORM 010-READ-TRANSACTION
               UNTIL WS-TRANSACTION-SWITCH = 'YES'.
           CLOSE TRANSACTION-FILE
                 VSAM-FILE.
           STOP RUN.
       010-READ-TRANSACTION.
           MOVE TR-ID-NUMBER    TO VSAM-ID-NUMBER.
           MOVE TR-NAME         TO VSAM-NAME.
           MOVE TR-SALARY       TO VSAM-SALARY.
           WRITE VSAM-MASTER-RECORD
               INVALID KEY DISPLAY
                 'RECORD OUT OF SEQUENCE OR DUPLICATE' TR-ID-NUMBER.
           READ TRANSACTION-FILE
               AT END MOVE 'YES' TO WS-TRANSACTION-SWITCH.
//GO.VSAMMAST DD DSN=V01BLUND.COBKS.CLUSTER,DISP=SHR
//GO.SYSOUT DD SYSOUT=*
//GO.SYSPRINT DD SYSOUT=*
//GO.SYSUDUMP DD SYSOUT=*
//GO.SYSIN DD *
A13496050LABELLE H.                     013600
A24963216MONROE S.                      020000
A37693774LALONDE T.                     035000
A44567891GOLD M.                        042100
A49725867MONROE S.                      017500
//* TO LIST KSDS AND DELETE IT
//LISTCL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT -
   INDATASET(V01BLUND.COBKS.CLUSTER) -
   CHARACTER
//COB2PGM  EXEC  COBVSCG
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. VSUPDATE.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-370.
       OBJECT-COMPUTER. IBM-370.
       INPUT-OUTPUT SECTION.
```

```
FILE-CONTROL.
    SELECT TRANSACTION-FILE ASSIGN TO UT-S-TRANS.
    SELECT ERROR-FILE ASSIGN TO UT-S-SYSOUT.
    SELECT VSAM-FILE ASSIGN TO VSAMMAST
        ACCESS IS RANDOM
        ORGANIZATION IS INDEXED
        RECORD KEY IS VSAM-ID-NUMBER.
DATA DIVISION.
FILE SECTION.
FD  VSAM-FILE
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 40 CHARACTERS
    DATA RECORD IS VSAM-RECORD.
01  VSAM-RECORD.
    05 VSAM-ID-NUMBER              PIC X(9).
    05 VSAM-NAME                   PIC X(25).
    05 VSAM-SALARY                 PIC X(6).
FD  TRANSACTION-FILE
    LABEL RECORDS ARE OMITTED
    RECORD CONTAINS 80 CHARACTERS
    DATA RECORD IS TRANS-RECORD.
01  TRANS-RECORD                   PIC X(80).
FD  ERROR-FILE
    LABEL RECORDS ARE OMITTED
    RECORD CONTAINS 133 CHARACTERS
    DATA RECORD IS ERROR-RECORD.
01  ERROR-RECORD                   PIC X(133).
WORKING-STORAGE SECTION.
77  WS-EOF-INDICATOR               PIC X(3)
                                   VALUE 'NO '.
77  WS-INVALID-SWITCH              PIC X(3).
    88 NO-MATCH-OR-NEW-RECORD          VALUE 'YES'.
01  WS-TRANS-RECORD.
    05 WS-TRANS-ID                 PIC X(9).
    05 WS-TRANS-NAME               PIC X(25).
    05 WS-TRANS-SALARY             PIC X(6).
    05 FILLER                      PIC X(39).
    05 WS-TRANS-CODE               PIC X.
       88 ADDITION          VALUE 'A'.
       88 DELETION          VALUE 'D'.
       88 SALARY-CHANGE     VALUE 'C'.
01  WS-ERROR-MESSAGE-1             PIC X(40)
                                   VALUE ' RECORD ALREADY IN FILE '.
01  WS-ERROR-MESSAGE-2             PIC X(40)
                                   VALUE ' NO MATCH '.
01  WS-PRINT-RECORD.
    05 WS-PRINT-MESSAGE            PIC X(40).
    05 WS-PRINT-ID                 PIC X(9).
    05 FILLER                      PIC X(5)
                                   VALUE SPACES.
    05 WS-PRINT-NAME               PIC X(25).
    05 FILLER                      PIC X(53)
                                   VALUE SPACES.
01  WS-VSAM-RECORD.
    05 WS-VSAM-ID                  PIC X(9).
    05 WS-VSAM-NAME                PIC X(25).
    05 WS-VSAM-SALARY              PIC X(6).
```

```
       PROCEDURE DIVISION.
           MOVE 'NO ' TO WS-INVALID-SWITCH.
           OPEN INPUT TRANSACTION-FILE
                I-O VSAM-FILE
                OUTPUT ERROR-FILE.
           READ TRANSACTION-FILE INTO WS-TRANS-RECORD
               AT END MOVE 'YES' TO WS-EOF-INDICATOR.
           PERFORM 010-READ-VSAM-FILE THRU 010-READ-VSAM-FILE-EXIT
               UNTIL WS-EOF-INDICATOR = 'YES'.
           CLOSE TRANSACTION-FILE
                 VSAM-FILE
                 ERROR-FILE.
           STOP RUN.
       010-READ-VSAM-FILE.
           MOVE WS-TRANS-ID TO VSAM-ID-NUMBER.
           READ VSAM-FILE INTO WS-VSAM-RECORD
               INVALID KEY MOVE 'YES' TO WS-INVALID-SWITCH.
           IF NO-MATCH-OR-NEW-RECORD
               PERFORM 070-NEW-RECORD THRU 070-NEW-RECORD-EXIT
           ELSE
               PERFORM 060-MATCH-ROUTINE THRU 060-MATCH-ROUTINE-EXIT.
           READ TRANSACTION-FILE INTO WS-TRANS-RECORD
               AT END MOVE 'YES' TO WS-EOF-INDICATOR.
       010-READ-VSAM-FILE-EXIT.
           EXIT.
       060-MATCH-ROUTINE.
           IF ADDITION
               MOVE WS-TRANS-NAME    TO WS-PRINT-NAME
               MOVE WS-TRANS-ID      TO WS-PRINT-ID
               MOVE WS-ERROR-MESSAGE-1 TO WS-PRINT-MESSAGE
               WRITE ERROR-RECORD FROM WS-PRINT-RECORD
               GO TO 060-MATCH-ROUTINE-EXIT.
           IF DELETION
               DELETE VSAM-FILE
               GO TO 060-MATCH-ROUTINE-EXIT.
           IF SALARY-CHANGE
               MOVE WS-TRANS-SALARY TO WS-VSAM-SALARY
               REWRITE VSAM-RECORD FROM WS-VSAM-RECORD.
       060-MATCH-ROUTINE-EXIT.
           EXIT.
       070-NEW-RECORD.
           MOVE 'NO ' TO WS-INVALID-SWITCH.
           IF ADDITION
               MOVE WS-TRANS-ID      TO WS-VSAM-ID
               MOVE WS-TRANS-NAME    TO WS-VSAM-NAME
               MOVE WS-TRANS-SALARY TO WS-VSAM-SALARY
               WRITE VSAM-RECORD FROM WS-VSAM-RECORD
           ELSE
               MOVE WS-TRANS-NAME    TO WS-PRINT-NAME
               MOVE WS-TRANS-ID      TO WS-PRINT-ID
               MOVE WS-ERROR-MESSAGE-2 TO WS-PRINT-MESSAGE
               WRITE ERROR-RECORD FROM WS-PRINT-RECORD.
       070-NEW-RECORD-EXIT.
           EXIT.
//GO.VSAMMAST DD DSN=V818LUND.COBKS.CLUSTER,DISP=SHR
//GO.SYSOUT DD SYSOUT=*
```

```
//GO.TRANS DD *
A103624/BSIMPSON V.                    030000                    A
A134960DOLABELLE P.                                              D
A20486/DSMILES C.                      020000                    A
A249652186INSTON J.                    010000                    A
A30999/645381SSEL W.                   030000                    A
A59E2146HPICARD T.                     050000                    A
A864328870WILEY S.                     015000                    C
//* TO LIST KSDS AND DELETE IT
//LIS EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT -
   INDATASET(V818LUND.COBKS.CLUSTER) -
   CHARACTER
 LISTCAT -
   CAT(EDUCUS.MASTRGUY) -
   ENT(V818LUND.COBKS.CLUSTER) -
   ALL
 DEFINE AIX -
   (NAME(V818LUND.COBKSCS.ALTINDEX) -
   VOLUME(EDUC05) -
   TRACKS(1 1) -
   KEYS(25 9) -
   RECORDSIZE(39 75) -
   NONUNIQUEKEY -
   UNIQUE -
   RELATE(V818LUND.COBKS.CLUSTER)) -
   CATALOG(EDUCUS.MASTRGUY)
//BUILD EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 BLDINDEX -
   INDATASET(V818LUND.COBKS.CLUSTER) -
   OUTDATASET(V818LUND.COBKSCS.ALTINDEX) -
   CATALOG(EDUCUS.MASTRGLY)
 DEFINE PATH -
   (NAME(V818LUND.COBKSCS.ALTPATH) -
   PATHENTRY(V818LUND.COBKSCS.ALTINDEX)) -
   CATALOG(EDUCUS.MASTRGLY)
 PRINT INDATASET (V818LUND.COBKSCS.ALTPATH) CHAR
 DELETE -
   V818LUND.COBKS.CLUSTER PURGE
```

Contents of KSDS After Execution of
COBOL Create Program

```
LISTING OF DATA SET -V818LUNC.COBKS.CLUSTER

KEY OF RECORD - A13496050
A13496050LABELLE M.                    013600

KEY OF RECORD - A24963216
A24963216MONROE S.                     020000

KEY OF RECORD - A37693774
A37693774LALONDE T.                    035000

KEY OF RECORD - A44567891
A44567891GOLD M.                       042100

KEY OF RECORD - A49725867
A49725867MONROE S.                     017500

IDC00051 NUMBER OF RECORDS PROCESSED WAS 5

IDC00001 FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Report Produced by
COBOL KSDS Update Program

```
RECORD ALREADY IN FILE          A24963216    WINSTON B.
NO MATCH                        A86458893    WILEY S.
```

Contents of KSDS after execution of
COBOL Update Program

```
LISTING OF DATA SET -V816LUNC.COBKS.CLUSTER

KEY OF RECORD - A1036247
A1036247SIMPSON V.                    030000

KEY OF RECORD - A2048655
A2048655MILES C.                      020000

KEY OF RECORD - A24963216
A24963216MONROE S.                    020000

KEY OF RECORD - A30999764
A30999764JONES W.                     030000

KEY OF RECORD - A3769374
A3769374LALONDE T.                    035000

KEY OF RECORD - A44567891
A44567891GOLD M.                      042100

KEY OF RECORD - A49725867
A49725867MONROE S.                    017500

KEY OF RECORD - A59621464
A59621464PICARD T.                    050000

IDC00051 NUMBER OF RECORDS PROCESSED WAS 8

IDC00011 FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Contents of KSDS via
Alternate Path ALTPATH

```
LISTING OF DATA SET -VBIGLUND.CUSKSDS.ALTPATH

KEY OF RECORD - GOLD P.
A44536789GOLD P.                    042100

KEY OF RECORD - LALONDE T.
A37693714LALONDE T.                 055000

KEY OF RECORD - MILES C.
A20406755MILES C.                   020000

KEY OF RECORD - MONROE S.
A24563216MONROE S.                  020000

KEY OF RECORD - MONROE S.
A49725867MONROE S.                  017500

KEY OF RECORD - PICARD T.
A59621464PICARD T.                  050000

KEY OF RECORD - SIMPSON V.
A10362478SIMPSON V.                 060000

KEY OF RECORD - SWISSEL W.
A30999764538135EL W.                060000

IDC00051 NUMBER OF RECORDS PROCESSED WAS 8

IDC00011 FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

Appendix C - Sample VSAM File Processing Program
COBOL RRDS Create
COBOL RRDS Sequential Update
COBOL RRDS Random Retrieval

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ 9GUE
//JOBCAT DD DSN=EDUCUS.MASTRGUY,DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE V818LUNO.COBRRDS.CLUSTER PURGE
 DEFINE CLUSTER -
    (NAME(V818LUNO.COBRRDS.CLUSTER) -
    VOL(EDUCUS) -
    NUMBERED -
    IRK(2 2) -
    RECS(105 105) -
    UNIQUE -
    FOR(30)) -
    CATALOG(EDUCUS.MASTRGUY)
//STEP2 EXEC COBVSCG,PARM='STATE,FLOW'
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CREATE.
       REMARKS. THIS PROGRAM CREATES A RELATIVE FILE OF SUMMARY SALES RECORDS
                USING SEQUENTIAL ACCESS.  EACH RECORD CONTAINS A 5-YEAR
                SUMMARY OF UNIT AND DOLLAR SALES FOR ONE WEEK OF THE
                YEAR; THERE ARE 52 RECORDS WITHIN THE FILE, EACH
                REPRESENTING 1 WEEK.
                EACH INPUT RECORD REPRESENTS THE SUMMARY SALES FOR ONE
                WEEK OF ONE YEAR.  THE RECORDS FOR THE FIRST WEEK OF
                THE LAST FIVE YEARS (IN ASCENDING ORDER) ARE THE FIRST
                FIVE INPUT RECORDS.  THE RECORDS FOR THE SECOND WEEK OF
                THE LAST FIVE YEARS ARE THE NEXT FIVE INPUT RECORDS, ETC.
                THUS FIVE INPUT RECORDS FILL ONE OUTPUT RECORD.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-370.
       OBJECT-COMPUTER. IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT RELATIVE-FILE ASSIGN TO RELMAST
             ACCESS SEQUENTIAL
             ORGANIZATION RELATIVE
             FILE STATUS RRDS-STATUS.
           SELECT INPUT-FILE ASSIGN TO UT-S-SYSIN
             FILE STATUS READER-STATUS.
       DATA DIVISION.
```

```
       FILE SECTION.
       FD  RELATIVE-FILE
           LABEL RECORDS ARE STANDARD.
       01  RELATIVE-RECORD.
           05  RELATIVE-RECORD-WEEKLY OCCURS 5.
               10  RELATIVE-YEAR                    PIC 99.
               10  RELATIVE-WEEK                    PIC 99.
               10  RELATIVE-UNIT-SALES              PIC S9(6).
               10  RELATIVE-DOLLAR-SALES            PIC S9(9)V99.
       FD  INPUT-FILE
           LABEL RECORDS ARE OMITTED.
       01  INPUT-RECORD.
           05  INPUT-YEAR                           PIC 99.
           05  INPUT-WEEK                           PIC 99.
           05  INPUT-UNIT-SALES                     PIC S9(6).
           05  INPUT-DOLLAR-SALES                   PIC S9(9)V99.
           05  FILLER                               PIC X(59).
       WORKING-STORAGE SECTION.
       77  I                       PIC 9.
       01  ERROR-STATUS.
           05  RROS-STATUS     PIC XX VALUE 'ZZ'.
           05  READER-STATUS   PIC XX VALUE 'ZZ'.
       01  OUT-OF-CARDS-FLAG   PIC XXX VALUE 'NO '.
           88  OUT-OF-CARDS            VALUE 'YES'.
       PROCEDURE DIVISION.
       MAIN-ROUTINE.
           PERFORM INITIALIZATION-ROUTINE.
           PERFORM READ-CREATE-ROUTINE UNTIL OUT-OF-CARDS.
           PERFORM TERMINATION-ROUTINE.
           STOP RUN.
       INITIALIZATION-ROUTINE.
           OPEN INPUT INPUT-FILE,
                OUTPUT RELATIVE-FILE.
           IF RROS-STATUS NOT = '00' OR READER-STATUS NOT = '00'
               MOVE 'YES' TO OUT-OF-CARDS-FLAG
               DISPLAY 'OPEN ERROR ', RROS-STATUS, READER-STATUS.
       READ-CREATE-ROUTINE.
           PERFORM READ-MOVE-ROUTINE VARYING I FROM 1 BY 1 UNTIL I > 5.
           IF NOT OUT-OF-CARDS
              WRITE RELATIVE-RECORD
              IF RROS-STATUS NOT = '00'
                 MOVE 'YES' TO OUT-OF-CARDS-FLAG
                 DISPLAY 'WRITE ERROR ', RROS-STATUS.
       READ-MOVE-ROUTINE.
           READ INPUT-FILE INTO RELATIVE-RECORD-WEEKLY (I)
             AT END MOVE 'YES' TO OUT-OF-CARDS-FLAG.
       TERMINATION-ROUTINE.
           CLOSE INPUT-FILE,
           IF RROS-STATUS NOT = '00' OR READER-STATUS NOT = '00'
               DISPLAY 'CLOSE ERROR ', RROS-STATUS, READER-STATUS.
//GO.RELMAST DD DSN=V810LOND.COBRROS.CLUSTER,DISP=SHR
//GO.SYSOUT DD SYSOUT=*
//GO.SYSOUT DD SYSOUT=*
```

```
//GO.SYSIN DD *
7501    300         12000
7601    196         97500
7701    410         81000
7801    259         49800
7901    257         45000
7502    120         12000
7602    201         62300
7702    410         71000
7802    102         20400
7902    205         41000
7503    482         48000
7603    150         54900
7703    210         61000
7803    965         93600
7903    261         46100
7504    150         51000
7604    192         59200
7704    193         51000
7804    274         58400
7904    347         60300
7505    170         62000
7605    178         47800
7705    153         41000
7805    410         82000
7905    425         91500
7506    190         84000
7606    300         60000
7706    150         31000
7806    459         95300
7906    481         91600
7507    210         96000
7607    292         59200
7707    126         21000
7807    615         98300
7907    222         44400
7508    250         74800
7608    193         79300
7708    476         98300
7808    533         99900
7908    128         28300
7509    350         17900
7609    175         85000
7709    253         97300
7809    342         69400
```

```
7646    424       92500
7746    425       98200
7846    107       29500
7946    281       56000
7545    350       70000
7645    524       78900
7745    050       11100
7845    113       21400
7945    223       46500
7550    410       82000
7550    193       67500
7750    111       22200
7850    149       21100
7950    226       67000
7551    420       64000
7651    173       81900
7751    123       33500
7851    211       62300
7951    283       77500
7552    359       71000
7652    294       72500
7752    153       44400
7852    242       42900
7952    294       98000
//STEP3 EXEC PGM=IDCAMS
//STSPRINT DD SYSOUT=*
//SYSIN DD *
  PRINT INDATASET(V818LUND.COBKRDS.CLUSTER) CHAR
//STEP4 EXEC COBVSC,PARM='STATE,FLOW'
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. UPDATERR.
       REMARKS.  THIS PROGRAM UPDATES THE FILE OF SUMMARY SALES RECORDS
                 PREVIOUSLY CREATED KRDS, USING SEQUENTIAL ACCESS.
                 THE UPDATING PROGRAM ADDS A RECORD FOR THE NEW YEAR, AND
                 DELETES THE OLDEST YEAR'S RECORDS FROM THE RELATIVE-FILE.
                THE INPUT RECORD REPRESENSIS THE SUMMARY SALES RECORD FOR ONE
                 WEEK OF THE LAST PRECEDING YEAR.  THE RELATIVE KEY FOR THE
                 RELATIVE-FILE IS PRESENT IN THE INPUT RECORD AS INPUT-WEEK.
                  WRITTEN.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-370.
       OBJECT-COMPUTER. IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT RELATIVE-FILE ASSIGN TO RELMAST
             ACCESS SEQUENTIAL
             ORGANIZATION RELATIVE
             RELATIVE KEY INPUT-WEEK
             FILE STATUS KRDS-STATUS.
           SELECT INPUT-FILE ASSIGN TO UT-S-SYSIN
             FILE STATUS READER-STATUS.
```

```
       DATA DIVISION.
       FILE SECTION.
       FD  RELATIVE-FILE
           LABEL RECORDS ARE STANDARD.
       01  RELATIVE-RECORD.
           05  FIRST-FOUR-YEARS            PIC X(84).
           05  LAST-YEAR                   PIC X(21).
       FD  INPUT-FILE
           LABEL RECORDS ARE OMITTED.
       01  INPUT-RECORD.
           05  INPUT-YEAR                  PIC 99.
           05  INPUT-WEEK                  PIC 99.
           05  INPUT-UNIT-SALES            PIC S9(6).
           05  INPUT-DOLLAR-SALES          PIC S9(9)V99.
           05  FILLER                      PIC X(59).
       WORKING-STORAGE SECTION.
       01  WORK-RECORD.
           05  FIRST-YEAR                  PIC X(21).
           05  LAST-FOUR-YEARS             PIC X(84).
       01  FILE-STATUS-AREA.
           05  RRDS-STATUS                 PIC XX VALUE '22'.
           05  READER-STATUS               PIC XX VALUE '22'.
       01  END-OF-FILE-FLAG                PIC X(3) VALUE 'NO '.
           88  END-OF-FILE                          VALUE 'YES'.
       PROCEDURE DIVISION.
       MAIN-ROUTINE.
           PERFORM INITIALIZATION-ROUTINE.
           PERFORM UPDATE-WEEK-ROUTINE UNTIL END-OF-FILE.
           CLOSE INPUT-FILE,
                 RELATIVE-FILE.
           STOP RUN.
       INITIALIZATION-ROUTINE.
           OPEN INPUT INPUT-FILE,
                I-O RELATIVE-FILE.
           IF READER-STATUS NOT = '00' OR RRDS-STATUS NOT = '00'
              DISPLAY 'OPEN ERROR ', READER-STATUS, RRDS-STATUS
              MOVE 'YES' TO END-OF-FILE-FLAG.
       UPDATE-WEEK-ROUTINE.
           READ INPUT-FILE AT END MOVE 'YES' TO END-OF-FILE-FLAG.
           IF NOT END-OF-FILE
             READ RELATIVE-FILE INTO WORK-RECORD AT END
              MOVE 'YES' TO END-OF-FILE-FLAG.
           IF READER-STATUS NOT = '00' OR RRDS-STATUS NOT = '00'
              DISPLAY 'READ STATUS ', READER-STATUS, RRDS-STATUS
              MOVE 'YES' TO END-OF-FILE-FLAG.
           IF NOT END-OF-FILE
              MOVE LAST-FOUR-YEARS TO FIRST-FOUR-YEARS
              MOVE INPUT-RECORD TO LAST-YEAR
              REWRITE RELATIVE-RECORD
              IF RRDS-STATUS NOT = '00'
                DISPLAY 'REWRITE ERROR ', RRDS-STATUS
                MOVE 'YES' TO END-OF-FILE-FLAG.
//GO.RELMAST DD DSN=VSAMCOBO.COBRRDS.CLUSTER,DISP=SHR
//GO.SYSOUT DD SYSOUT=*
//GO.SYSOUT DD SYSOUT=*
```

```
//GO.STSIN DD *
8001    456        81000
8002    678        91800
8003    789        92800
8004    890        99900
8005    123        52500
8006    234        46800
8007    132        26400
8008    125        25000
8009    350        70000
8010    600        90000
8011    610        91000
8012    620        92000
8013    630        93000
8014    640        94000
8015    650        78900
8016    660        66000
8017    710        88900
8018    720        98900
8019    856        99800
8020    923        92500
8021    713        91800
8022    614        81400
8023    693        93500
8024    695        79500
8025    125        22500
8026    225        42500
8027    325        62500
8028    425        72500
8029    525        82500
8030    625        85000
8031    725        89900
8032    625        87500
8033    925        92500
8034    252        50000
8035    352        75700
8036    453        75600
8037    554        95400
8038    655        95500
8039    756        95600
8040    857        85700
8041    958        95800
8042    123        23600
8043    223        44600
8044    333        66600
8045    433        93800
8046    533        53300
8047    633        93500
8048    733        91000
8049    149        22800
8050    289        48900
8051    334        67800
8052    410        82000
```

```
//STEP5 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  PRINT INDATASET(VSAMCUND.COBRRDS.CLUSTER) CHAR
//STEP6 EXEC COBVSCG,PARM='STATE,FLOW'
//COB.SYSIN DD *
       IDENTIFICATION DIVISION.
       PROGRAM-ID. RETRVE-RELATIVE.
       REMARKS.  THIS PROGRAM, USING DYNAMIC ACCESS, RETRIEVES THE RRDS
                 SUMMARY SALES FILE.
                 THE RECORDS OF THE INPUT-FILE CONTAIN ONE REQUIRED FIELD
                 (INPUT-WEEK) WHICH IS THE RELATIVE KEY FOR RELATIVE-FILE
                 AND ONE OPTIONAL FIELD (END-WEEK).
                 AN INPUT RECORD CONTAINING DATA IN INPUT-WEEK AND SPACES IN
                 END-WEEK REQUESTS A PRINTOUT FOR THAT ONE SPECIFIC
                 RELATIVE-RECORD; THE RECORD IS RETRIEVED THROUGH RANDOM
                 ACCESS.
                 AN INPUT RECORD CONTAINING DATA IN BOTH INPUT-WEEK AND END-WEE
                 REQUESTS A PRINTOUT OF ALL THE RELATIVE-FILE RECORDS WITHIN TH
                 RELATIVE KEY RANGE OF INPUT-WEEK THROUGH END-WEEK, INCLUSIVE;
                 THESE RECORDS ARE RETRIEVED THROUGH SEQUENTIAL ACCESS.
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SOURCE-COMPUTER. IBM-370.
       OBJECT-COMPUTER. IBM-370.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT RELATIVE-FILE ASSIGN TO RELMAST
             ACCESS DYNAMIC
             ORGANIZATION RELATIVE
             RELATIVE KEY INPUT-WEEK.
           SELECT INPUT-FILE ASSIGN TO UR-S-SYSIN.
           SELECT PRINT-FILE ASSIGN TO UR-S-SYSPRINT.
       DATA DIVISION.
       FILE SECTION.
       FD  RELATIVE-FILE
           LABEL RECORDS ARE STANDARD.
       01  RELATIVE-RECORD-01.
           05 RELATIVE-RECORD OCCURS 5.
              10 RELATIVE-YEAR          PIC 99.
              10 RELATIVE-WEEK          PIC 99.
              10 RELATIVE-UNIT-SALES    PIC 9(6).
              10 RELATIVE-DOLLAR-SALES PIC 9(9)V99.
       FD  INPUT-FILE
           LABEL RECORDS ARE OMITTED.
       01  INPUT-RECORD.
           05  INPUT-WEEK                    PIC 99.
           05  END-WEEK                      PIC 99 BLANK WHEN ZERO.
           05  FILLER                        PIC X(76).
       FD  PRINT-FILE
           LABEL RECORD ARE OMITTED.
       01  PRINT-RECORD.
           05  FILLER                        PIC X.
           05  PRINT-WEEK                    PIC 99.
           05  FILLER                        PIC X(5).
           05  PRINT-YEAR                    PIC 99.
           05  FILLER                        PIC X(3).
           05  PRINT-UNIT-SALES              PIC ZZZ,ZZ9.
           05  FILLER                        PIC X(5).
           05  PRINT-DOLLAR-SALES            PIC $$$,$$$,$$..99.
```

```
       WORKING-STORAGE SECTION.
       77  I                                PIC 9 VALUE ZERO.
       01  END-OF-FILE-FLAG                 PIC X(3) VALUE 'NO'.
           88  END-OF-FILE                           VALUE 'YES'.
       01  OUT-OF-CARDS-FLAG                PIC X(3) VALUE 'NO'.
           88 OUT-OF-CARDS                           VALUE 'YES'.
       PROCEDURE DIVISION.
       MAIN-ROUTINE.
           PERFORM INITIALIZATION-ROUTINE.
           PERFORM RETRIEVAL-ROUTINE UNTIL OUT-OF-CARDS.
           PERFORM TERMINATION-ROUTINE.
           STOP RUN.
       INITIALIZATION-ROUTINE.
           OPEN INPUT INPUT-FILE,
                      RELATIVE-FILE,
                OUTPUT PRINT-FILE.
           MOVE SPACE TO PRINT-RECORD.
       RETRIEVAL-ROUTINE.
           READ INPUT-FILE AT END MOVE 'YES' TO OUT-OF-CARDS-FLAG.
           IF NOT OUT-OF-CARDS
             IF END-WEEK = SPACES
               PERFORM RANDOM-PROCESS
             ELSE
               PERFORM SEQUENTIAL-PROCESS.
       RANDOM-PROCESS.
           READ RELATIVE-FILE RECORD
             INVALID KEY DISPLAY 'NO RECORD FOR WEEK ', INPUT-WEEK.
           PERFORM WRITE-WEEK-SALES VARYING I FROM 1 BY 1 UNTIL I > 5.
       WRITE-WEEK-SALES.
           MOVE RELATIVE-WEEK (I) TO PRINT-WEEK.
           MOVE RELATIVE-YEAR (I) TO PRINT-YEAR.
           MOVE RELATIVE-UNIT-SALES (I) TO PRINT-UNIT-SALES.
           MOVE RELATIVE-DOLLAR-SALES (I) TO PRINT-DOLLAR-SALES.
           WRITE PRINT-RECORD AFTER ADVANCING 2 LINES.
       SEQUENTIAL-PROCESS.
           START RELATIVE-FILE.
           MOVE 'NO' TO END-OF-FILE-FLAG.
           PERFORM READ-SEQUENTIAL UNTIL PRINT-WEEK > END-WEEK
             OR END-OF-FILE-FLAG = 'YES'.
       READ-SEQUENTIAL.
           READ RELATIVE-FILE NEXT RECORD
             AT END MOVE 'YES' TO END-OF-FILE-FLAG.
           IF NOT END-OF-FILE
             PERFORM WRITE-WEEK-SALES VARYING I FROM 1 BY 1 UNTIL I > 5.
       TERMINATION-ROUTINE.
           CLOSE INPUT-FILE,
                 RELATIVE-FILE,
                 PRINT-FILE.
//GO.SYSPRINT DD SYSOUT=*
//GO.SYSOUT DD SYSOUT=*
//GO.SYSDBOUT DD SYSOUT=*
//GO.RELMAST DD DSN=V818LUND.COBBKDS.CLUSTER,DISP=SHR
//GO.SYSIN DD *
09
1218
43
45
4952
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DELETE V818LUND.COBBKDS.CLUSTER PURGE
```

Contents of RRDS after Execution of COBOL Create Program

LISTING OF DATA SET -V818LUND.COBRRDS.CLUSTER

RELATIVE RECORD NUMBER - 1 7501 300	120007601	196	975007701	410	810007801	259	498007901	257	45000
RELATIVE RECORD NUMBER - 2 7502 120	120007602	201	625007702	410	710007802	102	204007902	205	41000
RELATIVE RECORD NUMBER - 3 7503 482	480007603	150	549007703	210	610007803	365	936007903	261	46100
RELATIVE RECORD NUMBER - 4 7504 150	510007604	192	592007704	193	510007804	274	584007904	347	60300
RELATIVE RECORD NUMBER - 5 7505 170	620007605	178	478007705	153	410007805	410	820007905	425	91300
RELATIVE RECORD NUMBER - 6 7506 190	840007606	300	600007706	150	310007806	459	953007906	481	91600
RELATIVE RECORD NUMBER - 7 7507 210	960007607	292	592007707	126	210007807	615	983007907	222	44400
RELATIVE RECORD NUMBER - 8 7508 250	748007608	193	795007708	476	983007808	333	999007908	128	28300
. . .									
RELATIVE RECORD NUMBER - 49 7549 350	700007649	528	769007749	050	111007849	113	214007949	225	46500
RELATIVE RECORD NUMBER - 50 7550 410	820007550	193	675007750	111	222007850	149	211007950	226	67000
RELATIVE RECORD NUMBER - 51 7551 420	840007651	178	819007751	123	333007851	211	623007951	283	77500
RELATIVE RECORD NUMBER - 52 7552 359	710007652	294	725007752	153	444007852	242	429007952	294	98000

Contents of RRDS after Execution of COBOL Sequential Update Program

RELATIVE RECORD NUMBER - 1 7601 196	975007701	410	81000/601	259	498007901	257	450008001	456	81000
RELATIVE RECORD NUMBER - 2 7602 201	623007702	410	71000/602	102	204007902	205	410008002	678	91800
RELATIVE RECORD NUMBER - 3 7603 150	549007703	210	61000/603	365	936007903	261	461008003	789	92800
RELATIVE RECORD NUMBER - 4 7604 192	592007704	193	51000/604	274	584007904	347	603008004	890	99900
RELATIVE RECORD NUMBER - 5 7605 178	478007705	153	41000/605	419	820007905	425	913008005	123	52300
RELATIVE RECORD NUMBER - 6 7606 300	600007706	150	31000/606	459	953007906	481	916008006	234	46800
RELATIVE RECORD NUMBER - 7 7607 292	592007707	126	21000/607	615	983007907	222	444008007	132	26400
RELATIVE RECORD NUMBER - 8 7608 193	793007708	476	98300/608	333	995007908	128	283008008	125	25000
				.					
				.					
				.					
				.					
RELATIVE RECORD NUMBER - 49 7649 328	789007749	050	11100/649	113	214007949	225	465008049	149	22800
RELATIVE RECORD NUMBER - 50 7650 193	675007750	111	22200/650	149	211007950	226	670008050	289	48900
RELATIVE RECORD NUMBER - 51 7651 173	819007751	123	33300/651	211	623007951	283	775008051	354	67800
RELATIVE RECORD NUMBER - 52 7652 294	725007752	153	44400/652	243	425007952	294	980008052	410	82000

Report Produced by
COBOL RRDS Random Retrieval Program

09	76	175	$850.00
09	77	254	$975.00
09	78	342	$694.00
09	79	481	$865.00
09	80	550	$700.00
12	76	292	$595.00
12	77	583	$945.00
12	78	259	$618.00
12	79	141	$284.00
12	80	620	$920.00
13	76	950	$760.00
13	77	482	$955.00
13	78	225	$450.00
13	79	119	$518.00
13	80	650	$950.00
14	76	450	$999.00
14	77	394	$925.00
14	78	169	$363.00
14	79	237	$464.00
14	80	640	$940.00
15	76	850	$698.00
15	77	465	$915.00
15	78	300	$600.00
15	79	355	$789.00
15	80	650	$789.00
16	76	119	$797.00

16	77	515	$903.00
16	78	144	$710.00
16	79	144	$710.00
16	80	660	$660.00
17	76	80	$545.00
17	77	428	$855.00
17	78	124	$800.00
17	79	481	$963.00
17	80	710	$887.00
18	76	91	$695.00
18	77	398	$885.00
18	78	584	$730.00
18	79	515	$715.00
18	80	720	$989.00
19	76	51	$955.00
19	77	592	$873.00
19	78	432	$862.00
19	79	407	$814.00
19	80	856	$998.00
43	76	132	$289.00
43	77	250	$563.00
43	78	514	$914.00
43	79	218	$835.00
43	80	225	$448.00
45	76	595	$875.00
45	77	101	$257.00
45	78	550	$842.00
45	79	220	$745.00

48	80	488	$956.00
49	76	368	$789.00
49	77	50	$111.00
49	78	113	$214.00
49	79	223	$463.00
49	80	149	$228.00
50	75	193	$673.00
50	77	111	$222.00
50	78	149	$219.00
50	79	226	$670.00
50	80	289	$489.00
51	76	173	$619.00
51	77	123	$333.00
51	78	211	$623.00
51	79	283	$775.00
51	80	354	$676.00
52	76	294	$725.00
52	77	153	$444.00
52	78	242	$429.00
52	79	294	$780.00
52	80	410	$820.00

Appendix D - Sample VSAM File Processing Program
PL/I ESDS Create and Extend
IDCAMS Creation of Alternate Index over ESDS
PL/I Alternate Index Reference of ESDS

SOURCE LISTING

```
*JOB CLASS=*
/*XEQ     9VUE
//JOBCAT DD DSN=EDUCOS.MASTRGUT,DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE V818LUNO.PLIESDS.CLUSTER PURGE
 DEFINE CLUSTER( -
    NAME(V818LUNO.PLIESDS.CLUSTER) -
    VOLUMES(EDUCOS) -
    NONINDEXED -
    RECORDSIZE(80 80) -
    TRACKS(2 2) -
    FOR(2) -
    UNIQUE) -
  CATALOG(EDUCOS.MASTRGUT)
//STEP2 EXEC PLIXCLG,PARM.PLI='NEST,NOESD,NOXREF,OBJECT,NODECK'
//PLI.SYSIN DD *
 CREATE:PROC OPTIONS(MAIN);

       DCL FAMFILE FILE SEQUENTIAL OUTPUT ENV(VSAM),
           IN FILE RECORD INPUT,
           STRING CHAR(80);
       DCL SYSPRINT FILE PRINT OUTPUT;
       DCL I FIXED BIN(15);

       ON ENDFILE(IN) GOTO FINITO;

       DO I=1 BY 1;
         READ FILE(IN) INTO (STRING);
         WRITE FILE(FAMFILE) FROM (STRING);
       END;

 FINITO:PUT SKIP EDIT(I-1,' RECORDS PROCESSED')(A);
 END;
//LKED.SYSLMOD DD DSN=V818LUNO.MYPROGS(ESCREATE),DISP=(NEW,PASS),
// VOL=SER=EDUCOS,UNIT=PRIVATE
//GO.SYSLMOD DD DUMMY
//GO.FAMFILE DD DSN=V818LUNO.PLIESDS.CLUSTER,DISP=SHR
//GO.IN DD *
FRED                            69          M
ANDY                            70          M
SUZAN                           72          F
```

```
//LIST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT INDATASET(V818LUND.PLIESUS.CLUSTER) CHARACTER
//STEP3 EXEC PGM=ESCREATE
//STEPLIB DD DSN=V818LUND.MYPROGS.DISP=SHR
//SYSPRINT DD SYSOUT=*
//FAMFILE DD DSN=V818LUND.PLIESUS.CLUSTER,DISP=SHR
//IN DD *
JANE                          75          F
//LIST EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 LISTCAT -
   CATALOG(EDUCOS.MASTRCAT) -
   ENT(V818LUND.PLIESUS.CLUSTER) -
   ALL
 PRINT INDATASET(V818LUND.PLIESUS.CLUSTER) CHARACTER
//* TO CREATE A UNIQUE KEY ALTERNATE INDEX FOR AN ESDS
//ALIX  EXEC  PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 DEFINE ALTERNATEINDEX( -
   NAME(V818LUND.PLIESUS.ALTINDEX) -
   VOLUMES(EDUCOS) -
   TRACKS(4 1) -
   KEYS(15 0) -
   RECORDSIZE(20 40) -
   UNIQUEKEY -
   UNIQUE -
   RELATE(V818LUND.PLIESUS.CLUSTER)) -
    CATALOG(EDUCOS.MASTRCAT)
//ALIX2  EXEC  PGM=IDCAMS
//DD1  DD  DSN=V818LUND.PLIESUS.CLUSTER,DISP=SHR
//DD2  DD  DSN=V818LUND.PLIESUS.ALTINDEX,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 BLDINDEX -
   INFILE(DD1) -
   OUTFILE(DD2) -
  CATALOG(EDUCOS.MASTRCAT)
 DEFINE PATH( -
   NAME(V818LUND.PLIESUS.ALTPATH) -
   PATHENTRY(V818LUND.PLIESUS.ALTINDEX)) -
  CATALOG(EDUCOS.MASTRCAT)
//* TO CREATE A NON-UNIQUE KEY ALTERNATE INDEX PATH FOR AN ESDS
//NONU  EXEC  PGM=IDCAMS
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 DEFINE ALTERNATEINDEX( -
   NAME(V818LUND.PLIESUS.SEXIND) -
   VOL(EDUCOS) -
   TRACKS(4 1) -
   KEYS(1 3/) -
   NONUNIQUEKEY -
   RELATE(V818LUND.PLIESUS.CLUSTER) -
   UNIQUE -
   RECORDSIZE(20 400)) -
  CATALOG(EDUCOS.MASTRCAT)
```

```
//NONE2  EXEC  PGM=IDCAMS
//DD1   DD  DSN=VB18LUNO.PLIESDS.CLUSTER,DISP=SHR
//DD2   DD  DSN=VB18LUNO.PLIESDS.SEXIND,DISP=SHR
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 BLDINDEX -
   INFILE(DD1) -
   OUTFILE(DD2) -
  CATALOG(EDUCOD.MASTRGUY)
 DEFINE PATH( -
   NAME(VB18LUNO.PLIESDS.SEXPATH) -
   PATHENTRY(VB18LUNO.PLIESDS.SEXIND)) -
  CATALOG(EDUCOD.MASTRGUY)
//* ALTERNATE INDEX PATHS AND BACKWARDS READING WITH AN ESDS
//LISTALT EXEC PLIXCG,PARM.PLI='NEST,CUSTMI,NOESD'
//PLI.SYSIN  DD  *
       /*  PROGRAM TO LIST ESDS USING ALTERNATE INDEXES  */
 READIT:PROC OPTIONS(MAIN);

       DCL CLUSFIL FILE SEQUENTIAL INPUT ENV(VSAM),
           BACKFIL FILE SEQUENTIAL INPUT ENV(VSAM BKWD),
           ALIXFIL FILE DIRECT INPUT ENV(VSAM),
           SEXFIL FILE KEYED SEQUENTIAL INPUT ENV(VSAM),
           SYSIN FILE STREAM INPUT,
           SYSPRINT FILE STREAM PRINT OUTPUT;


       DCL STRINGREC CHAR (80);
       DCL 1 STRUC DEF (STRINGREC),
             2 NAME       CHAR(25),
             2 BIRTHDATE  CHAR(2),
             2 FILL       CHAR(10),
             2 SEX        CHAR(1);

       DCL CARDNAME CHAR(25);
       DCL SAMEKEY BUILTIN;
       DCL (ENDCLUSFILE,ENDBACKFILE,ENDALIXFILE,ENDSEXFILE,ENDCARDS,
           KEYERROR) BIT(1) INIT('0'B);


       ON ENDFILE (CLUSFIL) ENDCLUSFILE='1'B;
       ON ENDFILE (BACKFIL)ENDBACKFILE='1'B;
       ON ENDFILE (ALIXFIL) ENDALIXFILE='1'B;
       ON ENDFILE (SEXFIL) ENDSEXFILE='1'B;
       ON KEY(ALIXFIL) KEYERROR='1'B;
       ON ENDFILE (SYSIN) ENDCARDS='1'B;


       /* READ CLUSTER ESDS FORWARDS*/

       PUT EDIT ('FAMILY ELDEST FIRST')(A);
       PUT SKIP(2);
       DO WHILE (^ENDCLUSFILE);
         READ FILE(CLUSFIL) INTO(STRINGREC);
         IF ^ ENDCLUSFILE THEN PUT SKIP EDIT (STRINGREC)(A);
       END;
       CLOSE FILE(CLUSFIL);  /* CLOSE NOT ESSENTIAL, BUT ADVISABLE*/
```

```
        /* READ CLUSTER ESDS BACKWARDS */

        PUT SKIP(3) EDIT('FAMILY YOUNGEST FIRST')(A);
        PUT SKIP(2);
        DO WHILE (^ENDBACKFILE);
          READ FILE(BACKFIL) INTO(STRINGREC);
          IF ^ ENDBACKFILE THEN PUT SKIP EDIT (STRINGREC)(A);
        END;
        CLOSE FILE(BACKFIL);


        /* READ CLUSTER ESDS USING UNIQUE ALTERNATE INDEX
           TO SELECT RECORDS DIRECTLY BY NAME  */

        /* PRINT DATE OF BIRTH OF CHILD SPECIFIED IN THE FILE SYSIN */


        PUT SKIP(3) EDIT('ALTERNATE INDEX BY NAME')(A);
        PUT SKIP(2);
        DO WHILE(^ENDCARDS);
          GET EDIT (CARDNAME)(COL(1),A(25));
          IF ^ENDCARDS THEN DO;
            READ FILE(ALIXFIL) INTO(STRINGREC) KEY(CARDNAME);
            IF ^ KEYERROR
                THEN PUT SKIP(2)EDIT(CARDNAME,'WAS BORN IN ',BIRTHDATE)
                  (A,X(1),A,A);
              ELSE PUT SKIP(2) EDIT (CARDNAME,
                  ' NOT A MEMBER OF THE SMITH FAMILY')(A);
            END;
        END;
        CLOSE FILE(ALIXFIL);


        /* READ SAME FILE USING NON-UNIQUE ALTERNATE INDEX */

        /* PRINT OUT ALL THE GIRLS IN THE FAMILY */

        PUT SKIP(3) EDIT ('ALL THE GIRLS')(A);
        PUT SKIP(2);
        READ FILE(SEXFIL) INTO (STRINGREC) KEY('F');
        PUT SKIP EDIT(STRINGREC)(A);
        DO WHILE(SAMEKEY(SEXFIL));
          READ FILE(SEXFIL) INTO(STRINGREC);
          PUT SKIP EDIT(STRINGREC)(A);
        END;
    END READIT;
//GO.CLUSFIL   DD  DSN=V818LUND.PLIESUS.CLUSTER,DISP=SHR
//GO.BACKFIL   DD  DSN=V818LUND.PLIESUS.CLUSTER,DISP=SHR
//GO.ALIXFIL   DD  DSN=V818LUND.PLIESUS.ALIPATH,DISP=SHR
//GO.SEXFIL   DD  DSN=V818LUND.PLIESUS.SEXPATH,DISP=SHR
//GO.SYSIN   DD  *
ANDY
FRED
TONY
//FIN EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
```

```
//SYSIN DD *
 PRINT INDATASET(V818LUND.PLIESUS.CLUSTER) CHAR
 PRINT -
   INDATASET (V818LUND.PLIESUS.ALTPATH) -
   CHAR
 PRINT -
   INDATASET (V818LUND.PLIESUS.SEXPATH) -
   CHAR
 DEL -
   V818LUND.PLIESUS.ALTINDEX -
   AIX -
   CAT(EDUCUS.MASTRGUY)
 DEL -
   V818LUND.PLIESUS.SEXIND -
   AIX -
   CAT(EDUCUS.MASTRGUY)
 DELETE V818LUND.PLIESUS.CLUSTER PURGE
```

Report Produced by PL/I ESDS Create Program

```
        3 RECORDS PROCESSED
```

Contents of ESDS after Execution of PL/I Create Program

```
LISTING OF DATA SET -V818LUNC.PLIESDS.CLUSTER

RBA OF RECORD = 0
FRED                    69          M

RBA OF RECORD = 80
ANDY                    70          M

RBA OF RECORD = 160
SUZAN                   72          F
```

Report Produced by PL/I ESDS Extend Program

```
        1 RECORDS PROCESSED
```

Contents of ESDS after Execution of PL/I Extend Program

```
LISTING OF DATA SET -V818LUNC.PLIESDS.CLUSTER

RBA OF RECORD = 0
FRED                    69          M

RBA OF RECORD = 80
ANDY                    70          M

RBA OF RECORD = 160
SUZAN                   72          F

RBA OF RECORD = 240
JANE                    75          F
```

Report Produced by
PL/I ESDS Alternate Index Reference Program

FAMILY ELDEST FIRST

FRED	69	M
ANDY	70	M
SUZAN	72	F
JANE	75	F

FAMILY YOUNGEST FIRST

JANE	75	F
SUZAN	72	F
ANDY	70	M
FRED	69	M

ALTERNATE INDEX BY NAME

ANDY	WAS BORN IN 70
FRED	WAS BORN IN 69
TONY	NOT A MEMBER OF THE SMITH FAMILY

ALL THE GIRLS

SUZAN	72	F
JANE	75	F

Contents of ESDS via
Alternate Pathway ALTPATH

```
LISTING OF DATA SET -V818LUND.PLIESUS.ALTPATH

KEY OF RECORD = ANDY
ANDY                    70          M

KEY OF RECORD = FRED
FRED                    69          M

KEY OF RECORD = JANE
JANE                    75          F

KEY OF RECORD = SUZAN
SUZAN                   72          F
```

Contents of ESDS via
Alternate Pathway SEXPATH

```
LISTING OF DATA SET -V818LUND.PLIESUS.SEXPATH

KEY OF RECORD = F
SUZAN                   72          F

KEY OF RECORD = F
JANE                    75          F

KEY OF RECORD = M
FRED                    69          M

KEY OF RECORD = M
ANDY                    70          M
```

Appendix E - Sample VSAM File Processing Program
PL/I KSDS Create
PL/I KSDS Update

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ     4WQUE
//JOBCAT DD DSN=EDUCUS.MASTRGUY,DISP=SHR
//* TO DEFINE AND LOAD A KSDS
//DEF EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE V818LUNG.PLIKSDS.CLUSTER PURGE
 DEFINE CLUSTER( -
   NAME(V818LUNG.PLIKSDS.CLUSTER) -
   VOLUMES(EDUCVS) -
   INDEXED -
   IMBED -
   REPLICATE -
   TRACKS(3 1) -
   KEYS(20 0) -
   RECORDSIZE(25 80) -
   UNIQUE) -
  CATALOG(EDUCUS.MASTRGUY)
//LOAD EXEC PLIXCG
//PLI.SYSIN DD *
 /* KSDS CREATE PROGRAM */
 TELNOS:PROC OPTIONS(MAIN);
  /*  PROGRAM TO CREATE A KEY SEQUENCED DATA SET  */

      DCL KSFIL FILE RECORD SEQUENTIAL KEYED ENV(VSAM),
          SYSIN FILE STREAM INPUT;
      DCL SYSPRINT FILE PRINT OUTPUT;

      DCL CARD   CHAR(80),
          NAME   CHAR(20) DEF CARD POS(1),
          NUMBER CHAR(5) DEF CARD POS(21),
          OUTREC CHAR(25) DEF CARD POS(1);
      DCL ENDCARDS BIT(1) INIT('0'B);

      ON ENDFILE(SYSIN) ENDCARDS='1'B;

      OPEN FILE(KSFIL) OUTPUT;

      DO WHILE (^ENDCARDS);
        GET EDIT(CARD)(A(80));
        IF ^ENDCARDS THEN WRITE FILE(KSFIL) FROM(OUTREC)
          KEYFROM(NAME);
      END;

      PUT EDIT('KSDS CREATION - PROCESSING COMPLETE')(A);
      CLOSE FILE(KSFIL);
 END TELNOS;
```

```
//GO.KSFIL DD DSN=VB18LUND.PLIKSUS.CLUSTER,DISP=SHR
//GO.SYSPRINT DD SYSOUT=*
//GO.SYSIN DD *
ACTION, G.          162
BAKER, K.           152
BRAMLEY, D. H.      248
CHEESEMAN, D.       141
CORT, G.            336
ELLIOTT, D.         875
FIGGINS, S.         413
HARVEY, C. D. W.    205
HASTINGS, G. A.     591
KENDALL, J. G.      294
LANCASTER, W.H.     624
MILES, K.           235
NEWMAN, M. W.       450
RITI, W. H.         515
ROLF, D. E.         114
SHEEKS, C. D.       241
SUTCLIFFE, A.       472
TAYLOR, G. C.       407
WILSON, L. W.       404
WINSTONE, E. A.     407
//PRI EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT -
   INDATASET(VB18LUND.PLIKSUS.CLUSTER) -
   CHARACTER
//* TO UPDATE A KSDS
//UP EXEC PLIXCG
//PLI.SYSIN DD *
 /* KSDS DIRECT UPDATE PROGRAM PLUS SEQUENTIAL LIST OF UPDATED FILE */
 DIRUPDT:PROC OPTIONS(MAIN);
      DCL KMAST FILE RECORD KEYED ENVIVSAM,
          SYSIN FILE STREAM INPUT,
          SYSPRINT FILE STREAM PRINT OUTPUT;

      DCL OUTREC CHAR(25),
          NUMBER CHAR(3)DEF OUTREC POS(21),
          NAME   CHAR(20) DEF OUTREC;
      DCL CODE CHAR(2),
          (ENDCARDS,ENDMAST) BIT(1) INIT('0'B);
      DCL UNCODE BUILTIN;

      ON ENDFILE (SYSIN)ENDCARDS='1'B;
      ON ENDFILE (KMAST) ENDMAST='1'B;
      ON KEY(KMAST) BEGIN;
        IF UNCODE=51 THEN PUT SKIP EDIT
          ('NOT FOUND:',NAME)(A(25),A);
        IF UNCODE=52 THEN PUT SKIP EDIT
          ('DUPLICATE:',NAME)(A(25),A);
      END;
```

```
/* UPDATE BEGINS HERE */

      OPEN FILE(KMAST) DIRECT UPDATE;
      DO WHILE (^ENDCARDS);
        GET EDIT(NAME,NUMBER,CODE)(A(20),A(5),X(56),A(1));
        IF ^ENDCARDS THEN DO;
          IF CODE='A' THEN WRITE FILE(KMAST) FROM (OUTREC)
            KEYFROM (NAME);
          ELSE IF CODE='C' THEN REWRITE FILE(KMAST)
            FROM(OUTREC) KEY (NAME);
          ELSE IF CODE ='D' THEN DELETE FILE(KMAST) KEY(NAME);
          ELSE PUT SKIP EDIT ('INVALID TRANSACTION CODE:',NAME)
            (A(25),A);
        END;
      END;
      CLOSE FILE(KMAST);


      /* SEQUENTIAL LIST TO VERIFY SUCCESSFUL UPDATE*/

      PUT PAGE;
      OPEN FILE(KMAST) SEQUENTIAL INPUT;
      DO WHILE (^ENDMAST);
        READ FILE(KMAST) INTO (OUTREC);
        IF^ENDMAST THEN PUT SKIP EDIT(OUTREC)(A);
      END;
      CLOSE FILE(KMAST);
  END;
//GO.KMAST DD DSN=V818LUND.PLIKSDS.CLUSTER,DISP=SHR
//GO.SYSPRINT DD SYSOUT=*
//GO.SYSIN DD *
NEWMAN, M. W.        516                                              C
GOODFELLOW, D. I.    889                                              A
MILES, K.                                                             D
HARVEY, C. D. W.     209                                              A
BARTLETT, S. G.      185                                              A
CORT, G.                                                              D
READ, K. M.          001                                              A
PITT, W. M.
ROLF, D. E.                                                           D
ELLIOTT, D.          291                                              C
HASTINGS, G. M.                                                       D
BRAMLEY, D. M.       439                                              C
//PR2 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  PRINT -
    INDATASET(V818LUND.PLIKSDS.CLUSTER) -
    CHARACTER
  LISTCAT -
    CAT(EDUCDS.MASTRCUY) -
    ENTRIES(V818LUND.PLIKSDS.CLUSTER) -
    ALL
  DEL V818LUND.PLIKSDS.CLUSTER PURGE
```

Contents of KSDS after Execution of
PL/I Create Program

```
LISTING OF DATA SET -V818LUNC.PLIKSDS.CLUSTER

KEY OF RECORD = ACTION, G.
ACTION, G.              162

KEY OF RECORD = BAKER, R.
BAKER, R.               152

KEY OF RECORD = BRAMLEY, O. H.
BRAMLEY, O. H.          248

KEY OF RECORD = CHEESEMAN, O.
CHEESEMAN, O.           141

KEY OF RECORD = CORY, G.
CORY, G.                336

KEY OF RECORD = ELLIOTT, O.
ELLIOTT, O.             875

KEY OF RECORD = FIGGINS, S.
FIGGINS, S.             413

KEY OF RECORD = HARVEY, C. O. W.
HARVEY, C. O. W.        205

KEY OF RECORD = HASTINGS, G. M.
HASTINGS, G. M.         391

KEY OF RECORD = KENDALL, J. G.
KENDALL, J. G.          294

KEY OF RECORD = LANCASTER, W.R.
LANCASTER, W.R.         624

KEY OF RECORD = MILES, R.
MILES, R.               233

KEY OF RECORD = NEWMAN, R. W.
NEWMAN, R. W.           450

KEY OF RECORD = PITT, W. H.
PITT, W. H.             515

KEY OF RECORD = ROLF, O. E.
ROLF, O. E.             114

KEY OF RECORD = SHEEKS, C. O.
SHEEKS, C. O.           241
```

KEY OF RECORD = SUTCLIFFE, M.
SUTCLIFFE, M. 472

KEY OF RECORD = TAYLOR, G. L.
TAYLOR, G. L. 407

KEY OF RECORD = WILTON, L. W.
WILTON, L. W. 404

KEY OF RECORD = WINSTONE, E. M.
WINSTONE, E. M. 307

Report Produced by PL/I KSDS Update Program

DUPLICATE:	HARVEY, C. D. W.
INVALID TRANSACTION CODE:	PITT, W. H.
NOT FOUND:	ROLF, D. E.

ACTION, G.	162
BAKER, K.	152
BARTLETT, S. G.	185
BRADLEY, D. H.	439
CHEESEMAN, D.	141
ELLIOTT, D.	291
FIGGINS, S.	415
GOODFELLOW, D. I.	889
HARVEY, C. D. W.	205
KENDALL, J. G.	294
LANCASTER, W.H	624
NEWMAN, M. W.	516
PITT, W. H.	515
READ, K. M.	001
ROLF, D. E.	114
SHEERS, C. D.	241
SUTCLIFFE, M.	472
TAYLOR, G. C.	407
WILTON, L. W.	404
WINSTONE, E. M.	507

Contents of KSDS after Execution of
PL/I Update Program

```
LISTING OF DATA SET -V818LUNC.PLIKSDS.CLUSTER

KEY OF RECORD = ACTION, G.
ACTION, G.          162

KEY OF RECORD = BAKER, K.
BAKER, K.           152

KEY OF RECORD = BARTLETT, S. G.
BARTLETT, S. G.     183

KEY OF RECORD = BRAMLEY, O. H.
BRAMLEY, O. H.      449

KEY OF RECORD = CHEESEMAN, O.
CHEESEMAN, O.       141

KEY OF RECORD = ELLIOTT, O.
ELLIOTT, O.         291

KEY OF RECORD = FIGGINS, S.
FIGGINS, S.         413

KEY OF RECORD = GOODFELLOW, D. I.
GOODFELLOW, D. I.   889

KEY OF RECORD = HARVEY, C. D. W.
HARVEY, C. D. W.    205

KEY OF RECORD = KENDALL, J. G.
KENDALL, J. G.      294

KEY OF RECORD = LANCASTER, W.R.
LANCASTER, W.R.     624

KEY OF RECORD = NEWMAN, M. W.
NEWMAN, M. W.       516

KEY OF RECORD = PITT, W. H.
PITT, W. H.         515

KEY OF RECORD = READ, K. M.
READ, K. M.         001

KEY OF RECORD = ROLF, D. E.
ROLF, D. E.         114

KEY OF RECORD = SHEEKS, C. D.
SHEEKS, C. D.       241
```

KEY OF RECORD = SUTCLIFFE, M.
SUTCLIFFE, M. 472

KEY OF RECORD = TAYLOR, G. L.
TAYLOR, G. L. 407

KEY OF RECORD = WILTON, L. W.
WILTON, L. W. 404

KEY OF RECORD = WINSTONE, E. M.
WINSTONE, E. M. 307

Appendix F - Sample VSAM File Processing Program
PL/I RRDS Create
PL/I RRDS Update

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ @@UE
//JOBCAT DD DSN=EDUCU5.MASTRGUY,DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE VB18LUND.PLIRRDS.CLUSTER PURGE
 DEFINE CLUSTER -
   (NAME(VB18LUND.PLIRRDS.CLUSTER) -
   VOL(EDUCU5) -
   NUMBERED -
   IRR(2 2) -
   RECS(20 20) -
   UNIQUE -
   FOR(5)) -
   CATALOG(EDUCU5.MASTRGUY)
//STEP2 EXEC PL1ACG,PARM='NEST,GOSTMT,NGESD'
//PLI.SYSIN DD *
 CRR1: PROC OPTIONS(MAIN);
       DCL RELFILE FILE RECORD OUTPUT DIRECT KEYED ENV(VSAM);
       DCL SYSIN FILE STREAM INPUT;
       DCL CARD CHAR(80),
           NAME CHAR(20) DEF CARD,
           NUMBER CHAR(2) DEF CARD POS(21);
       ON ENDFILE (SYSIN) GO TO FINISH;
       OPEN FILE(RELFILE);
 NEXT: GET FILE(SYSIN) EDIT(CARD)(A(80));
       WRITE FILE(RELFILE) FROM(NAME) KEYFROM(NUMBER);
       GO TO NEXT;
 FINISH: CLOSE FILE(RELFILE);
 END CRR1;
//GO.RELFILE DD DSN=VB18LUND.PLIRRDS.CLUSTER,DISP=SHR
//GO.SYSIN DD *
ACTION,G.          12
BAKER,K.           13
BRAMLET,D.M.       28
CHEESNAME,L.       11
CORT,G.            56
ELLIOTT,D.         65
FIGGINS,E.S.       43
HARVEY,L.D.W.      25
HASTINGS,G.M.      51
KENDALL,J.G.       24
LANCASTER,W.R.     64
```

```
FILES,M.              25
NEWMAN,M.J.           40
PITT,A.M.             35
POLF,D.E.             14
SHEEKS,C.D.           21
SURCLIFFE,M.          42
TAYLOR,G.C.           47
WILTON,L.J.           44
WINSTONE,E.M.         37
//STEP3 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 LISTCAT ENTRIES(V818LUNG.PLIKRDS.CLUSTER) ALL
 PRINT INDATASET(V818LUNG.PLIKRDS.CLUSTER) CHAR
//STEP4 EXEC PLIXCG,PARM='NEST,GOSTMT,NOESD'
//PLI.SYSIN DD *
                /* PROGRAM TO UPDATE A KRDS */
 ACK1:PROC OPTIONS(MAIN);

      DCL RELFILE FILE RECORD KEYED ENV(VSAM),
          SYSIN FILE STREAM INPUT,
          SYSPRINT PRINT OUTPUT,
          NAME CHAR(20),
          (NEWNO,OLDNO) CHAR(2),
          CODE CHAR(1),
          DUFIELD CHAR(20),
          BYTE CHAR(1) DEF DUFIELD;

      ON ENDFILE(SYSIN) GO TO PRINT;
      OPEN FILE(RELFILE) DIRECT UPDATE;
      ON KEY(RELFILE) PUT SKIP EDIT('DUPLICATE',NAME)(A(15),A);


      /* UPDATE THE KRDS DIRECTLY BY RELATIVE RECORD NUMBER */


 NEXT: GET EDIT(NAME,NEWNO,OLDNO,CODE)(A(20),2 A(2),X(55),A(1));
      IF CODE='A' THEN GO TO WRITE;
      ELSE IF CODE='C'
        THEN DO;
          DELETE FILE(RELFILE) KEY(OLDNO);
          GO TO WRITE;
        END;
      ELSE IF CODE='D'
        THEN DELETE FILE(RELFILE) KEY(OLDNO);
      ELSE PUT SKIP EDIT('INVALID CODE:',NAME)(A(5),A);
      GO TO NEXT;
 WRITE:WRITE FILE(RELFILE) KEYFROM(NEWNO) FROM (NAME);
      GO TO NEXT;


      /* PRINT OUT CONTENTS OF KRDS IN SEQUENTIAL ORDER */

 PRINT:CLOSE FILE(RELFILE);
      PUT PAGE;
      OPEN FILE(RELFILE) SEQUENTIAL INPUT;
      ON ENDFILE(RELFILE) GO TO FINISH;
```

```
NEXTIN:READ FILE(RELFILE) INTO (IOFIELD) KEYTO(NEWNO);
       PUT SKIP EDIT(NEWNO,IOFIELD)(A(5),A);
       GO TO NEXTIN;
FINISH:CLOSE FILE(RELFILE);
END ACK1.
//GO.RELFILE DD DSN=VBIBLUND.PLIPROG.CLUSTER,DISP=SHR
//GO.SYSIN DD *
NEWMAN,M.A.          5640
GOODFELLOW,D.T.      89
MILLS,K.               23
MARVEY,C.J.K.        24
MARTLETT,S.G.        13
CORT,G.                36
READ,K.M.            01
PITT,W.M.              55
ROLF,D.F.              14
ELLIOTT,D.           4285
MASTERS,G.M.           31
BRAMLEY,D.M.         4926
//STEP3 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 LISTCAT ENTRIES(VBIBLUND.PLIPROG.CLUSTER) ALL
 PRINT INDATASET(VBIBLUND.PLIPROG.CLUSTER) CHAR
 DELETE VBIBLUND.PLIPROG.CLUSTER PURGE
```

Contents of RRDS after Execution of
PL/I Create Program

```
LISTING OF DATA SET -VBIRLUNC.PLIRRDS.CLUSTER

RELATIVE RECORD NUMBER = 11
CHEESNAME.L.

RELATIVE RECORD NUMBER = 12
ACTION.G.

RELATIVE RECORD NUMBER = 13
BAKER.K.

RELATIVE RECORD NUMBER = 14
KULP.D.E.

RELATIVE RECORD NUMBER = 21
SHEERS.L.J.

RELATIVE RECORD NUMBER = 23
MILES.K.

RELATIVE RECORD NUMBER = 24
KENDALL.J.G.

RELATIVE RECORD NUMBER = 25
HARVEY.L.D.W.

RELATIVE RECORD NUMBER = 28
BRAMLEY.D.M.

RELATIVE RECORD NUMBER = 31
HASTINGS.G.M.

RELATIVE RECORD NUMBER = 36
CURT.G.

RELATIVE RECORD NUMBER = 37
WINSTONE.E.M.

RELATIVE RECORD NUMBER = 40
NEWMAN.M.W.

RELATIVE RECORD NUMBER = 42
SUTCLIFFE.M.

RELATIVE RECORD NUMBER = 43
FIGGINS.E.S.

RELATIVE RECORD NUMBER = 44
WILTON.L.W.
```

RELATIVE RECORD NUMBER = 54
TAYLOR,G.C.

RELATIVE RECORD NUMBER = 55
PITT,W.M.

RELATIVE RECORD NUMBER = 64
LANCASTER,W.R.

RELATIVE RECORD NUMBER = 65
ELLIOTT,D.

Report Produced by PL/I RRDS Update Program

```
DUPLICATE        BARTLETT,S.G.
INVALID CODE:    PITT,W.M.
DUPLICATE        ELLIOTT,D.
```

```
 1   READ,R.M.
11   CHEESNAKE,L.
12   ACTION,G.
13   BAKER,R.
21   SHEENS,C.D.
24   KENDALL,J.G.
25   HARVEY,C.D.W.
29   HARVEY,C.D.W.
37   WINSTONE,E.M.
42   SURCLIFFE,M.
43   FIGGINS,E.S.
44   WILTON,L.W.
47   TAYLOR,G.C.
49   BRAMLEY,D.M.
55   PITT,W.M.
56   NEWMAN,M.W.
64   LANCASTER,W.R.
89   GOODFELLOW,D.T.
```

Contents of RRDS after Execution of
PL/I Update Program

```
LISTING OF DATA SET -VSIGLUNC.PLIRRDS.CLUSTER

RELATIVE RECORD NUMBER = 1
READ,R,M.

RELATIVE RECORD NUMBER = 11
CHEESNAME,L.

RELATIVE RECORD NUMBER = 12
ACTION,G.

RELATIVE RECORD NUMBER = 15
BAKER,R.

RELATIVE RECORD NUMBER = 21
SHEERS,L,D.

RELATIVE RECORD NUMBER = 24
KENDALL,J,G.

RELATIVE RECORD NUMBER = 25
HARVEY,L,D,W.

RELATIVE RECORD NUMBER = 29
HARVEY,L,D,W.

RELATIVE RECORD NUMBER = 37
WINSTONE,E,M.

RELATIVE RECORD NUMBER = 42
SURCLIFFE,M.

RELATIVE RECORD NUMBER = 43
FIGGINS,E,S.

RELATIVE RECORD NUMBER = 44
WILTON,L,W.

RELATIVE RECORD NUMBER = 47
TAYLOR,G,C.

RELATIVE RECORD NUMBER = 49
BRAMLEY,D,M.

RELATIVE RECORD NUMBER = 55
PITT,W,M.

RELATIVE RECORD NUMBER = 56
NEWMAN,M,W.
```

```
RELATIVE RECORD NUMBER = 64
LANCASTER,W.H.

RELATIVE RECORD NUMBER = 65
GOODFELLOW,D.I.
```

Appendix G - Sample VSAM File Processing Program
Assembler ESDS Create and Extend

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ 8QUE
//JOBCAT DD DSN=EDUCUS.MASTRGUY,DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE V818LUND.ASMESDS.CLUSTER PURGE
 DEFINE CLUSTER -
   (NAME(V818LUND.ASMESDS.CLUSTER) -
   IRK(1 1) -
   NONINDEXED -
   FOR(5) -
   VOL(EDUCUS) -
   RECSZ(60 60) -
   UNIQUE) -
   CATALOG(EDUCUS.MASTRGUY)
//STEP2 EXEC ASMGCL
//ASM.SYSIN DD *
          PRINT NOGEN
START     BEGIN
          OPEN  (READER,INPUT)
          OPEN  (PRINTER,OUTPUT)
          OPEN  ESDSFIL
LOOP      GET   READER,CARD
          PUT   RPL=LIST
          LTR   15,15
          BNZ   ERROR
          B     LOOP
EOF1      CLOSE (READER,,PRINTER)
          CLOSE ESDSFIL
          STOP
ERROR     PUT   PRINTER,MSG
          B     EOF1
CARD      DS    CL80
MSG       DC    CL40'ERROR DURING WRITE TO ESDS'
ESDSFIL   ACB   MACRF=(AUR,SEQ,OUT),                              X
                DDNAME=ESDS,AM=VSAM
LIST      RPL   ACB=ESDSFIL,AREA=CARD,RECLEN=60,                  X
                AREALEN=60,OPTCD=(AUR,SEQ,MVE)
READER    DCB   DSORG=PS,LRECL=80,BLKSIZE=80,                     X
                RECFM=F,MACRF=GM,DDNAME=CARDS,EODAD=EOF1
PRINTER   DCB   DSORG=PS,MACRF=PMC,DDNAME=SYSPRINT,               X
                LRECL=100,BLKSIZE=100,RECFM=F
          END
//LKED.SYSLMOD DD DSN=A818LUND.MYPROGS(ESCREATE),
// DISP=(NEW,PASS),VOL=SER=LIBPU2,UNIT=STORAGE
//STEP2A EXEC PGM=*.STEP2.LKED.SYSLMOD
```

```
//CARDS DD *
SMITH
JONES
THOMPSON
MELLTEK
//ESDS DD DSN=V818LUND.ASMESDS.CLUSTER,DISP=SHR
//STSPRINT DD STSOUT=*
//STEP3 EXEC PGM=IDCAMS
//STSPRINT DD STSOUT=*
//STSIN DD *
 PRINT INDATASET(V818LUND.ASMESDS.CLUSTER) CHAR
//STEP4 EXEC PGM=*.STEP2.LKED.STSLMOD
//STSPRINT DD STSOUT=*
//ESDS DD DSN=V818LUND.ASMESDS.CLUSTER,DISP=SHR
//CARDS DD *
MYERS
JOHNSTON
//STEP5 EXEC PGM=IDCAMS
//STSPRINT DD STSOUT=*
//STSIN DD *
 PRINT INDATASET(V818LUND.ASMESDS.CLUSTER) CHAR
 DELETE V818LUND.ASMESDS.CLUSTER PURGE
```

Contents of ESDS after Execution of
Assembler Create Program

```
LISTING OF DATA SET -V818LUND.ASMESDS.CLUSTER

RBA OF RECORD = 0
SMITH

RBA OF RECORD = 60
JONES

RBA OF RECORD = 120
THOMPSON

RBA OF RECORD = 180
HELLYER
```

Contents of ESDS after Execution of
Assembler Extend Program

```
LISTING OF DATA SET -V818LUND.ASMESDS.CLUSTER

RBA OF RECORD = 0
SMITH

RBA OF RECORD = 60
JONES

RBA OF RECORD = 120
THOMPSON

RBA OF RECORD = 180
HELLYER

RBA OF RECORD = 240
AYERS

RBA OF RECORD = 300
JOHNSTON
```

Appendix H - Sample VSAM File Processing Program
Assembler KSDS Create

SOURCE LISTING

```
*JOB CLASS=Z
/*XEQ 3081E
//JOBCAT DD DSN=EDUC05.MASTRGUY.DISP=SHR
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DEFINE CLUSTER -
    (NAME(VS18LUNO.ASMKSDS.CLUSTER) -
    CYL(1 1) -
    INDEXED -
    FOR(5) -
    IMBED -
    VOL(EDUC05) -
    RECSZ(60 78) -
    KEYS(4,20) -
    UNIQUE) -
    CATALOG(EDUC05.MASTRGUY)
//STEP2 EXEC ASMFCG
//ASM.SYSIN DD *
          PRINT NOGEN
START     BEGIN
          OPEN   (READER,INPUT)
          OPEN   (PRINTER,OUTPUT)
          OPEN   KSDSFIL
LOOP1     GET    READER,CARD
          MVC    VREFNO,REFNO
          MVC    VAUTHOR,AUTHOR
          MVC    VTITLE,TITLE
          LA     9,L'VTITLE-1
          LA     8,1
          LA     6,1
          LA     3,VTITLE
LOOP2     LA     5,0(6,3)
          CLI    0(5),C''''
          BE     DONE
          BXLE   6,8,LOOP2
DONE      LA     3,25(6)
          MODCB  RPL=LIST,RECLEN=(3)
          LTR    15,15
          BNZ    ERROR
          PUT    RPL=LIST
          LTR    15,15
          BNZ    ERROR
          B      LOOP1
```

```
EOF      CLOSE (READER,,PRINTER)
         CLOSE KSDSFIL
         STOP
ERROR    PUT   PRINTER,MSG
         B     EOF
CARD     DS    0CL80
REFNO    DS    CL4
AUTHOR   DS    CL20
TITLE    DS    CL56
VBOOK    DS    0CL78
VAUTHOR  DS    CL20
VREFNO   DS    CL4
VTITLE   DS    CL54
MSG      DC    CL121'ERROR DURING WRITE TO KSDS OR MODCB'
KSDSFIL  ACB   MACRF=(OUT),DDNAME=KSDS
LIST     RPL   ACB=KSDSFIL,AREA=VBOOK,AREALEN=78
READER   DCB   DSORG=PS,LRECL=80,BLKSIZE=80,                    X
               RECFM=F,MACRF=GM,DDNAME=CARDS,EODAD=EOF
PRINTER  DCB   DSORG=PS,MACRF=PMC,DDNAME=SYSPRINT,              X
               LRECL=120,BLKSIZE=120,RECFM=F
         END   START
//GO.KSDS DD DSN=V818LUND.ASMKSDS.CLUSTER,DISP=SHR
//GO.SYSPRINT DD SYSOUT=*
//GO.CARDS DD *
1015W.SHAKESPEARE          'MUCH ADO ABOUT NOTHING'
1214L.CARROLL              'THE HUNTING OF THE SNARK'
3079G.FLAUBERT             'MADAME BOVARY'
3083V.M.HUGO               'LES MISERABLES'
3085J.K.JEROME             'THREE MEN IN A BOAT'
4285W.LANGLAND             'THE BOOK OF CERNING PIERS THE PLAWMAN'
5999O.KHAYYAM              'THE RUBAIYAT OF OMAR KHAYYAM'
6591F.RABELAIS             'THE HEROIC DEEDS OF GARGANTUA AND PANTAGRUEL'
8362H.D.THOREAU            'WALDEN, OR LIFE IN THE WOODS'
9765H.G.WELLS              'THE TIME MACHINE'
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 PRINT INDATASET(V818LUND.ASMKSDS.CLUSTER) CHAR
 DELETE V818LUND.ASMKSDS.CLUSTER PURGE
```

Contents of KSDS after Execution of
Assembler Create Program

```
LISTING OF DATA SET -V818LUND.ASMKSDS.CLUSTER

KEY OF RECORD - 1015
W.SHAKESPEARE          1015'MUCH ADO ABOUT NOTHING'

KEY OF RECORD - 1214
L.CARROLL              1214'THE HUNTING OF THE SNARK'

KEY OF RECORD - 3079
G.FLAUBERT             3079'MADAME BOVARY'

KEY OF RECORD - 3083
V.M.HUGO               3083'LES MISERABLES'

KEY OF RECORD - 3085
J.K.JEROME             3085'THREE MEN IN A BOAT'

KEY OF RECORD - 4295
W.LANGLAND             4295'THE BOOK CONCERNING PIERS THE PLOWMAN'

KEY OF RECORD - 5999
O.KHAYYAM              5999'THE RUBAIYAT OF OMAR KHAYYAM'

KEY OF RECORD - 6591
F.RABELAIS             6591'THE HEROIC DEEDS OF GARGANTUA AND PANTAGRUEL'

KEY OF RECORD - 8362
H.D.THOREAU            8362'WALDEN, OR LIFE IN THE WOODS'

KEY OF RECORD - 9765
H.G.WELLS              9765'THE TIME MACHINE'
```

BIBLIOGRAPHY

IBM Introduction to Direct Access Storage Devices and Organization Methods, Student Text

IBM Planning for Enhanced VSAM Under OS/VS, GC26-3842

IBM OS/VS2 Access Method Services, GC26-3841

IBM OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838

IBM VS COBOL for OS/VS, GC26-3857

IBM OS/VS COBOL Compiler and Library Programmer's Guide SC28-6483

IBM OS PL/I Checkout and Optimizing Compilers: Language Reference Manual, GC33-0009

IBM OS PL/I Optimizing Compiler: Programmer's Guide SC33-0006

IBM OS/VS Message Library: VS2 System Messages GC38-1002

SIMEQ, Gouvernement du Quebec: "Manuel d'exploitation" (1979)

International Data Corporation: Survey entitled "IBM Software Environment", cited in Computerworld, March 30, 1981 issue

Innovation Data Processing: "Innovation Access Method Product Description", Innovation, New Jersey (1979)

Gauer, Robert T. and Crawford, Marshal A. : "The COBOL Environment", Prentice-Hall (1979)